NightProbe User's Guide

CONCURRENT
COMPUTER
CORPORATION™

0890465-060
June 2004

The information contained in this document is believed to be correct at the time of publication. It is subject to change without notice. Concurrent Computer Corporation makes no warranties, expressed or implied, concerning the information contained in this document.

To report an error or comment on a specific portion of the manual, photocopy the page in question and mark the correction or comment on the copy. Mail the copy (and any additional comments) to Concurrent Computer Corporation, 2881 Gateway Drive, Pompano Beach, FL 33069-4324. Mark the envelope **"Attention: Publications Department**." This publication may not be reproduced for any other reason in any form without written permission of the publisher.

Printed in U. S. A.

Revision History:	Level:	Effective With:
Original Release -- November 1994	000	NightProbe 1.0
Current Release -- June 2004	060	NightProbe 2.7

Preface

Scope of Manual

This guide is designed to assist you in getting started with use of NightProbe, a real-time NightStar™ tool that provides a graphical user interface to the data recording services.

Structure of Manual

This manual consists of fifteen chapters and five appendices. A brief description of the chapters and appendixes is presented as follows.

- Chapter 1 introduces you to the concepts and components of NightProbe, a real-time tool that is part of the NightStar development environment.
- Chapter 2 explains how to invoke NightProbe.
- Chapter 3 introduces the components of NightProbe's Main window, the main control window for NightProbe.
- Chapter 4 describes the NightProbe's Target System Selection window.
- Chapter 5 describes the NightProbe's Timing Source Selection window.
- Chapter 6 describes the NightProbe's Output Selection window.
- Chapter 7 describes NightProbe's Program window.
- Chapter 8 describes NightProbe's PCI Device window.
- Chapter 9 describes the NightProbe's Shared Memory window.
- Chapter 10 describes the NightProbe's Mapped Memory window.
- Chapter 11 describes how to use the NightProbe's Item Browser window.
- Chapter 12 describes how to use the NightProbe's Item Definition window.
- Chapter 13 describes the NightProbe's Item Properties window.
- Chapter 14 describes in detail the NightProbe Spreadsheet Viewer window.
- Chapter 15 documents the NightProbe Application Programming Interface and provides sample programs to demonstrate usage of the data structures and functions in the API.

- Appendix A describes the syntax for specifying full variable names and eligilbility rules.
- Appendix B provides a reference table for keyboard shortcuts to activating menu items, controlling data sampling, and traversing dialogs.
- Appendix C consists of tutorials for probing a program and a PCI device.
- Appendix D contains information about customizing the NightProbe graphical user interface.
- Appendix E provides an overview of the primary factors that need to be taken into account prior to installing and running NightProbe on PowerMAX OS^{TM}.

Syntax Notation

The following notation is used throughout this manual:

italic Titles of books, reference cards, and items that you must specify appear in *italic* type. Special terms may also appear in *itali*cs.

`list bold` User input appears in **`list bold`** type and must be entered exactly as shown. Names of directories, files, commands, options and system manual page references also appear in **`list bold`** type.

`list` Operating system and program output such as prompts and messages and listings of files and programs appear in `list` type.

window Keyboard sequences and window features such as button, field, and menu labels, and window titles appear in window type.

`[]` Brackets enclose command options and arguments that are optional. You do not type the brackets if you choose to specify such options or arguments.

Referenced Publications

The following publications are referenced in this document:

0890285	*Real-Time Documentation Set*
0890300	*X Window System^{TM} User's Guide*
0890380	*OSF/Motif^{TM} Documentation Set*
0890398	*NightTrace^{TM} Manual*
0890429	*System Administration Volume 1*
0890430	*System Administration Volume 2*
0890423	*PowerMAX OS Programmer's Guide*
0890458	*NightSim^{TM} User's Guide*
0891055	*Élan^{TM} License Manager Release Notes*
0898004	*RedHawk Linux User's Guide*

Contents

Chapter 1 Overview

Chapter 2 Invoking NightProbe

Chapter 3 Main Window

Chapter 4 Target System Selection Window

Chapter 5 Timing Source Configuration

Chapter 6 Output Configuration

Chapter 7 Program Window

Chapter 8 PCI Device Window

Chapter 9 Shared Memory Window

Chapter 10 Mapped Memory Window

Chapter 11 Item Browser Window

Chapter 12 Item Definition Window

Chapter 13 Item Properties Window

Chapter 14 Spreadsheet Viewer

Chapter 15 NightProbe API

Appendix A Variables

Appendix B Keyboard Traversal

Appendix C Tutorials

Appendix D GUI Customization

Appendix E Target System Requirements

Index

Illustrations

Tables

1
Overview

1
Overview

NightProbe is a graphical tool for recording, viewing, and modifying data within a variety of resources:

- Executing Programs
- Shared Memory Segments
- Memory-Mapped Files and Devices
- PCI devices

Data is sampled using non-intrusive techniques to guarantee short response time and minimal impact on the target resources and the target system. The source code of target programs does not need to be modified or recompiled in order to be monitored. Executing programs can be monitored and recorded without being stopped and restarted.

Recording and Monitoring

Data recording refers to sampling memory locations in target resources and recording that data for subsequent analysis. Memory locations may be identified by logical address, offset, or by variable name. NightProbe allows you to record data to a file in NightProbe API format as well as NightTrace data format.

Data monitoring refers to displaying the sampled data for interactive visual inspection and, optionally, modification.

Eligible Variables

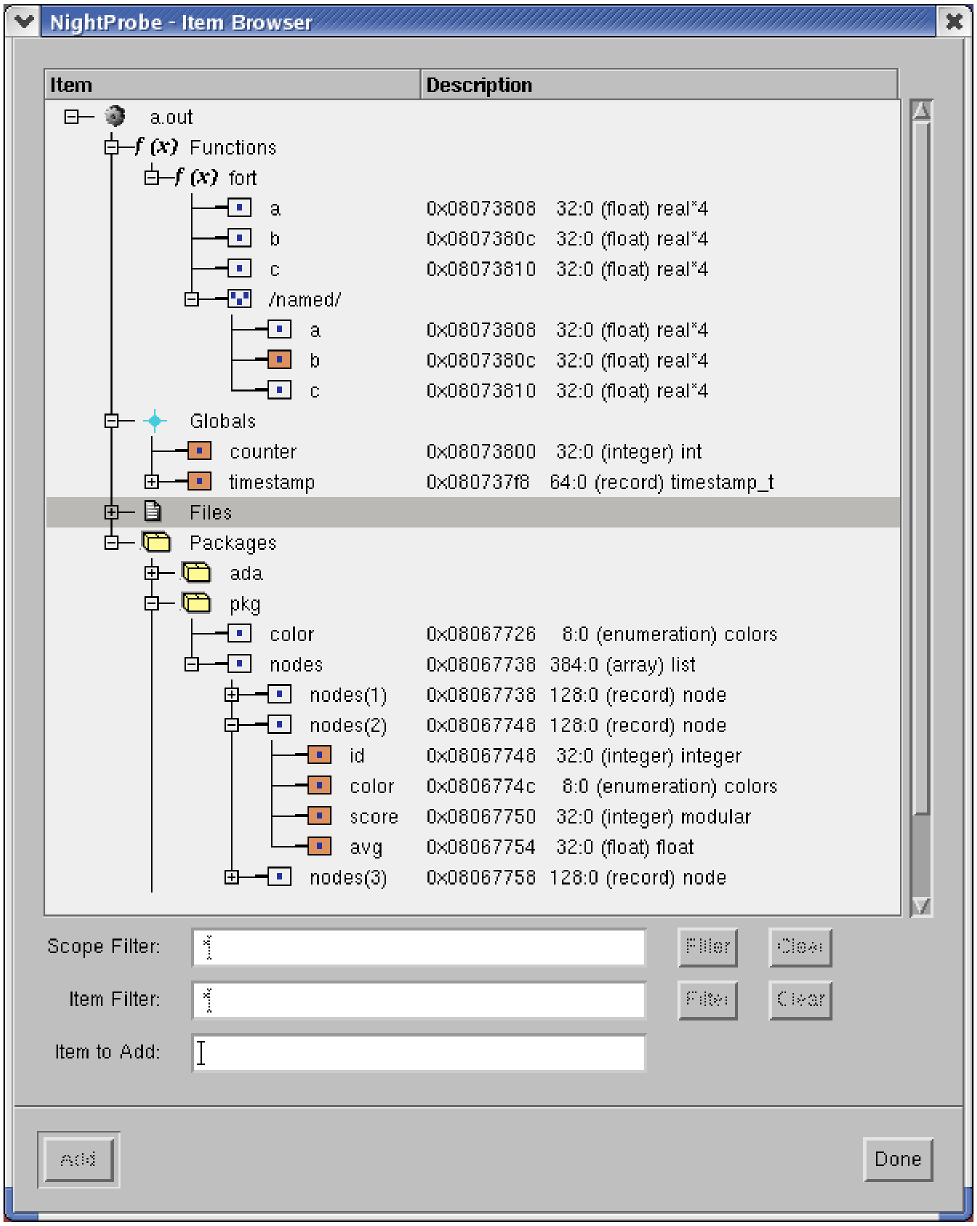
Variables with static addresses and layouts may be sampled. Supported languages include C, Fortran, and Ada programs built with the MAXAda compiler.

You can monitor and record any valid memory location in a program's address space, whether or not that location corresponds to a named variable or not.

NightProbe can be run on a different processor or system from the target resource, which minimizes NightProbe's impact the impact upon the target system.

Furthermore, NightProbe can probe executable programs which have been stripped of debug and symbol information, such as those in deployed scenarios. In such cases, a copy of the program containing the necessary debug and symbol information must be available for reading (not necessarily on the target system).

Variables are selected from program symbol files using a graphical tree browsing dialog as shown in Figure 1-1 :



Item	Description
a.out	
f (x) Functions	
f (x) fort	
a	0x08073808 32:0 (float) real*4
b	0x0807380c 32:0 (float) real*4
c	0x08073810 32:0 (float) real*4
/named/	
a	0x08073808 32:0 (float) real*4
b	0x0807380c 32:0 (float) real*4
c	0x08073810 32:0 (float) real*4
Globals	
counter	0x08073800 32:0 (integer) int
timestamp	0x080737f8 64:0 (record) timestamp_t
Files	
Packages	
ada	
pkg	
color	0x08067726 8:0 (enumeration) colors
nodes	0x08067738 384:0 (array) list
nodes(1)	0x08067738 128:0 (record) node
nodes(2)	0x08067748 128:0 (record) node
id	0x08067748 32:0 (integer) integer
color	0x0806774c 8:0 (enumeration) colors
score	0x08067750 32:0 (integer) modular
avg	0x08067754 32:0 (float) float
nodes(3)	0x08067758 128:0 (record) node

Figure 1-1. Selecting Variables with the Item Browser

Configuration files can be created and saved to retain variable selections and display layout, allowing for fast start-up on subsequent invocations of NightProbe.

Sampling Rates

Data sampling is the process of collecting the values from target memory locations and making the sampled data available to output or viewer processes that can record the data or display the data for interactive monitoring.

NightProbe supports three mechanisms which define the sampling rate:

- On-Demand sampling
- Iterative sampling driven from the system clock
- Iterative sampling controlled by a Frequency-Based Scheduler

Recording and Monitoring Procedures

The basic tasks relating to use of NightProbe include:

1. Define the target system
2. Select the target resources you wish to probe
3. Select variables from target resource programs or create artificial variables to create views into a target resource
4. Specify the desired sampling rate mechanism
5. Select Output methods
6. Connect NightProbe to the target system and target resources
7. Start Sampling

Privilege Requirements

On RedHawk Linux systems, you must have appropriate file access to the target resource or be granted the CAP_SYS_RAWIO capability in order to record, monitor, or modify the resource.

See Appendix E for information on the CAP_SYS_RAWIO capability.

2
Invoking NightProbe

2
Invoking NightProbe

nprobe

The NightProbe tool is available on your system as **`/usr/bin/nprobe`**. The format for executing **`nprobe`** is as follows:

To get information about NightProbe, use

```
    nprobe [--help] [--version]
```

or, to use NightProbe to record or monitor variable locations, use

```
    nprobe  [--record=output_file]
            [--trace=output_file]
            [-sheet=config_file]
            [--program=program_file]
            [--list] [-Xoption ...] [config_file | executable_file ]
```

or, to translate data files to text, use

```
    nprobe --input=data_file
```

Options are described as follows:

`--help`
This option allows you to display the usage information for **`nprobe`** and then exit. The X Window interface will not be started if you use this option.

`--version`
This option allows you to display the version and copyright information for **`nprobe`** and then exit. The X Window interface will not be started if you use this option.

`--program=`*progfile*
This option sends sampled data to the executable program *progfile* which is launched when sampling begins. The program should consume the data using the NightProbe API.

`--record=`*output_file*
Activate recording to file *output_file* when sampling begins.

`--trace=`*output_file*
Activate recording in NightTrace format to trace-event file *output_file* when sampling begins. An event map will be written to *output_file***`.evtmap`**.

`--sheet=`*layout_file*
Activate a spreadsheet viewer using the layout configuration in file *layout_file*.

`--list`
Activate a list viewer.

--input=*data_file*

Translate a previously recorded data file from its internal format to printable ASCII text. The **-i** option must be followed by the name of the data file. The text translation will be written to standard output. No other options should be used with **-i**. The X Window interface will not be started when using the **-i** option.

-*Xoption*

You may also specify any standard X Toolkit command-line option. Such options include **-bg** *color* to set the color for the window background; **-fg** *color* to set the color to use for text or graphics; and **-xrm** *resourcestring* to set selected resources. For a complete list of these options, refer to the **X(1)** system manual page.

config_file

This argument allows you to specify the name of a file that contains data sampling configuration data. You may specify a full or relative path name. The file may be one that you have created by using **nprobe** or a text editor of your choice.

If you use **nprobe** to create this file, you open a Main window, configure a data recording session, and then select the File Ì Save Session As menu item. Procedures are fully explained in Chapter 3, “Main Window”.

executable_file

This argument allows you to specify the name of an executable file. It is automatically added to the list of target resources so that you may immediately begin browsing for items within that file.

You may invoke **nprobe** without specifying any options or arguments.

nprobe requires that you have your `DISPLAY` environment variable set appropriately.

Getting Help

In addition to the *NightProbe User’s Guide*, there are several sources of information on the operation of NightProbe. These include:

- the *NightProbe Release Notes*
- the **nprobe(1)** and **nprobe(9)** system manual pages
- the **-help** command line option (described in “nprobe” on page 2-1)
- the Help menu on the menu bars of several of the NightProbe windows (see “Help” on page 3-14)
- the Help button on several of the NightProbe dialogs

  When you click on the Help button in one of these windows, the help information for that window is displayed

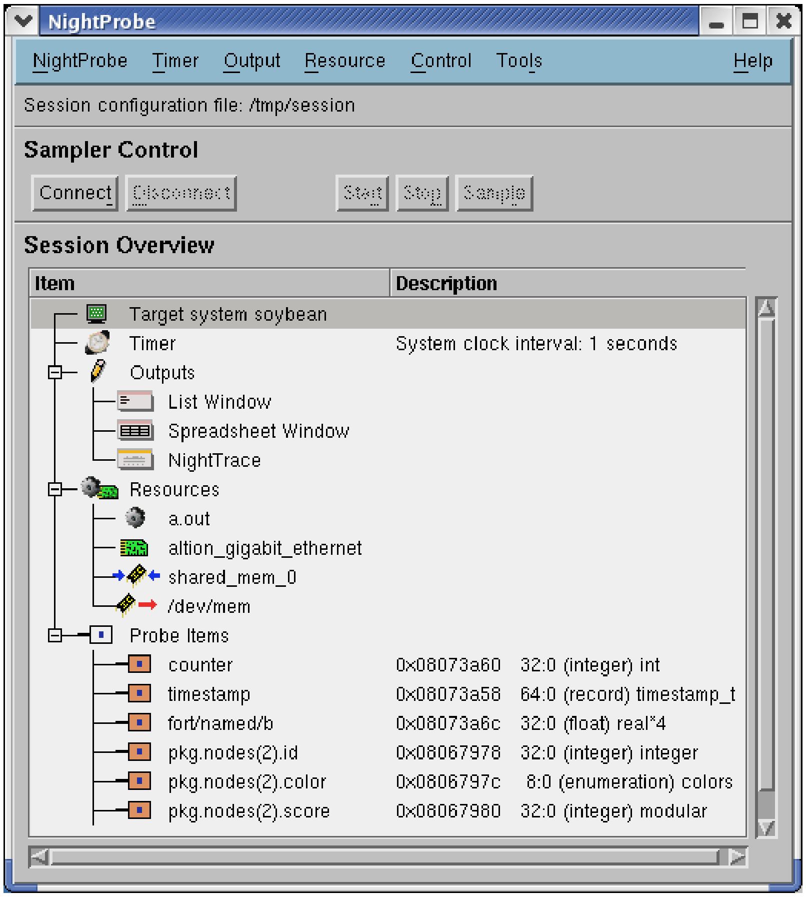
3
Main Window

3
Main Window

The Main window is the primary control window for NightProbe. From this window you will configure and control the data sampling process.

The Main window consists of the following components:

- The Session Overview Area
- The Sampler Control Area
- The Session Configuration Status Area
- The Menu Bar



Session configuration file: /tmp/session

Sampler Control

Session Overview

Item	Description
Target system soybean	
Timer	System clock interval: 1 seconds
Outputs	
List Window	
Spreadsheet Window	
NightTrace	
Resources	
a.out	
altion_gigabit_ethernet	
shared_mem_0	
/dev/mem	
Probe Items	
counter	0x08073a60 32:0 (integer) int
timestamp	0x08073a58 64:0 (record) timestamp_t
fort/named/b	0x08073a6c 32:0 (float) real*4
pkg.nodes(2).id	0x08067978 32:0 (integer) integer
pkg.nodes(2).color	0x0806797c 8:0 (enumeration) colors
pkg.nodes(2).score	0x08067980 32:0 (integer) modular

Figure 3-1. The Main Window

Session Overview Area

The Session Overview area provides primary access and control over the five main configurable areas of data recording:

- Target System Selection
- Timer Selection
- List of Outputs
- List of Target Resources
- List of Probe Items

All icons in the session overview area support right-click pop-up menu actions which allow you to add, remove, view, and manipulate the items.

These menus are not available when in the connected state; the session configuration may not be modified while connected.

In addition, doulbe click may be used to access properties where applicable.

Target System Selection

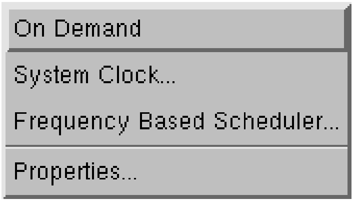
To the right of the Target System icon, the name of the target system is displayed.

Right-clicking the icon will display a menu which contains Properties.... Selection of that menu option launches the Target System Selection dialog which allows you to change the target system, the user name, and the scheduling attributes of the sampler process. This dialog may also be launched by double-clicking the Target System icon. See "Target System Selection Window" on page 4-1 for more information.

Timer Selection

To the right of the Timer icon, the description of the selected timing source is displayed.

Right-clicking the Timer icon displays the following menu:



By default, On Demand sampling is selected.

Selecting the On Demand option from the menu means the sampler will sample the variables only when the Sample button in the Sampler Control area is pressed.

Selecting the System Clock... option from the menu launches the System Timing Source Configuration window which allows you to chose the rate at which sampling will occur. See “System Clock Timing Source Configuration” on page 5-1for more information.

Selecting the Frequency Based Scheduler... option from the menu launches the FBS Timing Source Configuration window which allows you to chose the scheduler ID and period at which samples should be taken. Use of a frequency-based scheduler as the timing source allows for synchronization of data sampling with the target application, if that application is also scheduled on the same frequency-based scheduler. See “FBS Timing Source Configuration” on page 5-2 for more information.

Selecting the Properties... option from the Timer option menu will launch the appropriate timing source dialog so that you can make adjustments.

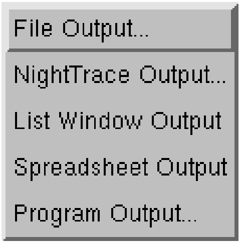
Selection of a timing source may also be initiated using the Timer menu on the menu bar of the Main window.

The timing source may not be changed while the sampler is in the connected state. The Timer option menus will be disabled while connected.

Outputs

The Outputs icon lists all outputs you select. The list can be expanded or collapsed by clicking the control box to the left of the icon.

Right-clicking the icon will display the following menu:



Selecting File Output... will launch the Recording File Specification window which allows you to specify a pathname for the file to which data samples will be written. See “File Specification” on page 6-1 for more information. A file icon and the name of the selected file will be added to the list of Outputs.

Selecting NightTrace Output... will launch the NightTrace Specification window which allows you to configure how NightTrace will capture and possibly display the data. See “NightTrace Specification” on page 6-2 for more information. An icon representing NightTrace will be added to the list of Outputs.

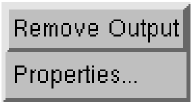
Selecting List Window Output will pop up a List Viewer window and add a List Window icon to the list of Outputs. The list viewer displays the names and values of all

Probe Items in a scrollable text area. See “List Window” on page 6-6 for more information.

Selecting Spreadsheet Output will pop up the Spreadsheet Viewer window and add a Spreadsheet Window icon to the list of outputs. The Spreadsheet Viewer displays variables in a configurable grid and allows for modification of variables as well. See “Spreadsheet” on page 6-9 for more information.

Multiple outputs can be added to the list, but only one instance of each output option can be present at any given time.

Once an item is added to the output list, right-clicking the icon representing it will display the following menu:



Selecting the Remove Output menu option removes the selected item from the list.

Selecting the Properties... menu item launches the appropriate output properties window which allows you to modify configurable settings.

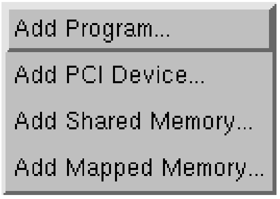
Selection of an output may also be initiated using the Output menu on the menu bar of the Main window.

Items in the Outputs list can not be removed nor their properties modified while in the connected state. While connected, the output menus are disabled.

Resources

The Resources icon lists all target resources you select. The list can be expanded or collapsed by clicking the control box to the left of the icon.

Right-clicking the icon will display the following menu:



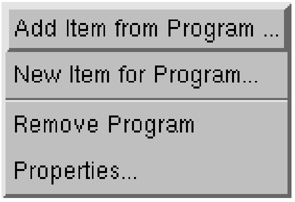
Selection of the Add Program menu option launches the Program Window which allows you to specify a program file, process name, and process ID. See “Program Window” on page 7-1 for more information.

Selection of the Add PCI Device menu option launches the PCI Device window which allows you to specify a PCI device which you wish to probe. The device may be specified using the Vendor ID, Device ID, Region Number, and Slot Number or can be selected from a descriptive list. See "PCI Device Window" on page 8-1 for more information.

Selection of the Add Shared Memory menu option launches the Shared Memory window which allows you to select a shared memory segment which you wish to probe. The segment may be selected by IPC Key, IPC Shared memory ID, IPC Key File pathname, or POSIX Shared Memory name. See "Shared Memory Window" on page 9-1 for more information.

Selection of the Add Mapped Memory menu option launches the Mapped Memory window which allows you to specify a device pathname which will be mapped and probed. The device may be **`/dev/mem`**, another device, or any file that the **`mmap(2)`** system service supports mapping. See "Mapped Memory Window" on page 10-1 for more information.

Once a resource is added to the output list, right-clicking the icon representing it will display a menu similar to the following:



Selecting the appropriate Remove menu option removes the selected resource from the list. Any variables associated with the resource will also be removed from the Probe Items list.

Selecting the Properties... menu option launches the appropriate resource properties window which allows you to modify configurable settings.

Selecting the Add Item From Program... menu option launches the Item Browser window which allows you to chose variables that you want to probe in that program. This menu option is not available for non-program resources. See "Item Browser Window" on page 11-1 for more information.

Selection of the appropriate New Item... menu option launches the Item Definition window which allows you to create artificial variables by browsing program files for types, selecting a desired type, and specifying resource addresses or offsets to the new items. These new items are not allocated by NightProbe in the target resource, they merely represent a view of a location that already exists in the resource. See Chapter 12, "Item Definition Window" for more information.

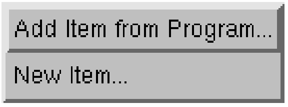
Selection of a resource may also be initiated using the Resource menu on the menu bar of the Main window.

Resources can not be removed nor their properties modified while in the connected state. While connected, the resource menus are disabled.

Probe Items

The Probe Items icon lists all target probe items you select in the order in which they will be sampled. The list can be expanded or collapsed by clicking the control box to the left of the icon.

Right-clicking the icon will display the following menu:

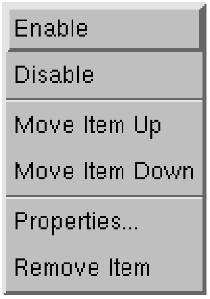


Selection of the Add Item from Program ... menu option launches the Item Browser window which allows you to select variables from program files and add them to the list of items to be probed. See "Item Browser Window" on page 11-1 for more information.

Selection of the appropriate New Item... menu option launches the Item Definition window which allows you to create artificial variables by browsing program files for types, selecting a desired type, and specifying resource addresses or offsets to the new items. These new items are not allocated by NightProbe in the target resource, they merely represent a view of a location that already exists in the resource. See Chapter 12, "Item Definition Window" for more information.

Once an item is added to the Probe Items list, its name and description will displayed to the right of the icon representing the item. The description includes the address, bit size, bit offset, the class of type, and the type name or type declaration.

Right-clicking on an item icon will display the following menu:



Selection of the Enable or Disable menu options toggles the enabled setting. If an item in the list is disabled, it is ignored and not included in data samples. Items may become disabled automatically if they are no longer valid with respect to their program file (e.g. a variable entered from a previous configuration file was since removed from the program). Enabled items are presented with an orange item icon while disabled icons are white. Multiple items may be disabled or enabled together using multiple selection.

Selection of the Move Item Up or Move Item Down menu options moves the item in the selected direction. The ordering of items in the list controls the order in which they are sampled. In most cases, this ordering is unimportant, but for some hardware devices the order in which items are referenced is critical. Items may also be reordered by selecting an item and using Shift-UpArrow and Shift-DownArrow keys.

Selection of the Properties... menu option launches the Item Properties window which allows you to select the default display format and specify an array slice for array items.

Selection of the Remove Item menu option removes the item from the Probe Items list. Multiple items may be removed together using multiple selection.

Item menus are disabled while in the connected state.

Sampler Control Area

The Sampler Control area provides buttons for controlling the state of data recording and monitoring as well text fields indicating its current status.

Connect
Accelerator: Ctrl-T

Pressing Connect initiates sampler activity. A sampler process is initiated on the target system. All memory pages associated with the selected variables for each resource are mapped into the sampler process's address space. Output methods and timer functions are elaborated which may involve creating or opening files, initiating NightTrace API calls, or joining a Frequency Based Scheduler. If any of these activities cannot be successfully completed, a diagnostic window is shown, all associations with target resources and the target system are released, and the connection process terminates.

You must be in a connected state to begin data sampling. Note that sampling does not actually begin until Start or Sample is pressed.

Changes to the session configuration are not allowed while in the connected state.

Disconnect
Accelerator: Ctrl-D

Pressing Disconnect terminates sampler activity. If sampling is active, it is stopped. The connection to the target system is terminated after all memory mappings to all target resources are released.

Start
Accelerator: Ctrl-R

Pressing Start initiates sampling. The Start button will be desensitized unless Connect has already been pressed and the selected timing source is either System Clock or Frequency Based Scheduler. If the timing source is System Clock, sampling will begin at the specified rate.

If the timing source is Frequency Based Scheduler, sampling will begin on the next cycle associated with the scheduler. If the scheduler is not running, sampling will not begin until the scheduler is started or resumed. The scheduler is controlled externally from NightProbe, by either NightSim, the **rtcp(1)** command, or a program using the Frequency Based Scheduler API (see **fbsconfigure(3)**). NightSim can be launched from the Tools menu

If NightTrace output has been selected, samples will be discarded if an associated NightTrace daemon is not running. They will begin to be collected when the daemon initiates. A NightTrace daemon can be launched via the **ntraceud(1)** command or from the NightTrace graphical interface which can be launched from the Tools menu.

Stop
Accelerator: Ctrl-P

Pressing Stop halts sampling. The Stop button will be desensitized unless Start has already been pressed. The sampler process remains in a connected state. Sampling will resume when Start is pressed. Note that if the selected timing source is the Frequency Based Scheduler, pressing Stop does <u>not</u> stop the scheduler -- it merely halts sampling. This will not result in scheduler overruns, as the sampler process continues to cycle, but without sampling activity.

Sample
Accelerator: Ctrl-L

Pressing Sample causes a single sample to be obtained from the sampler process. The Sample button will be desensitized unless Connect has already been pressed and the selected timing source is On Demand.

Session Configuration Status Area

The Session Configuration Status Area is located just below the menu bar and contains information on the name of the currently selected session configuration file and a warning indicator icon if the current configuration is different from what was loaded from or last saved to that file. Configuration files are selected with the Open Session... menu option on the File menu or are provided on the command line.

Menu Bar

The menu bar provides access to session configuration services, launching additional tools, and obtaining help. The activities provided in the pop-up menus in the Session

Overview area are available from the menu bar as well. The menu bar provides the following menus:

- NightProbe
- Timer
- Output
- Resource
- Control
- Tools
- Help

Each menu is described in the sections that follow.

NightProbe

Mnemonic: Alt-N

The NightProbe menu allows you to load a configuration, save the current configuration to a file, or create a new configuration. The NightProbe menu also contains the means to exit NightProbe.

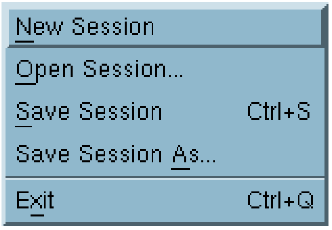


Figure 3-2. File menu

The following paragraphs describe the options on the File menu in more detail.

New Session
Mnemonic: N

> This option allows you to clear all information from the current session and reset the various areas to blank or default values. If the window contains unsaved changes, NightProbe displays a warning dialog. You may save the changes, clear the window without saving the changes, cancel the operation, or display help related to the dialog.

Open Session...
Mnemonic: O

This option allows you to open a sampler configuration file that you have previously saved and load all the configuration items into the current session.

If the window contains unsaved changes, NightProbe displays a warning dialog. You may proceed to open the new session, thereby discarding any unsaved changes, or cancel the operation.

When you select this option, NightProbe displays a file selection dialog.

To select the file to be opened, use the directory mask text area, scrolled list of directories, scrolled list of files, and file selection text area as appropriate. After making a selection, you may open the selected file, search for another file, cancel the operation, or display help related to the dialog.

Save Session

Mnemonic: S
Accelerator: <Control><S>

This option allows you to save the configuration data from the current session in the configuration file that is associated with the window. If the window is not associated with a configuration file name, this option is the same as Save Config File As.

Save Session As...

Mnemonic: A

This option allows you to specify the name of the file in which you wish the configuration data from the current session to be saved.

When you select this option, NightProbe displays a file selection dialog. After making a selection, you may save the configuration data from the current session in the selected file, search for another file, cancel the operation, or display help related to the dialog.

Exit

Mnemonic: X
Accelerator: <Control><Q>

This option exits NightProbe. If there are unsaved changes in the current session, a dialog will ask you if you wish to save the session before exiting.

Timer

Mnemonic: Alt-T

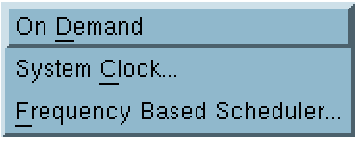


Figure 3-3. Timer menu

These menu items activate the timing selection as described in “Timer Selection” on page 3-2.

Output

Mnemonic: Alt-O

You must select at least one destination for output or NightProbe will not allow connection to the target resource (see “Sampler Control Area” on page 3-7).

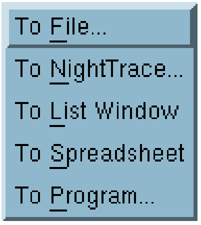


Figure 3-4. Output menu

These menu items activate output selection as described in “Outputs” on page 3-3.

Resource

Mnemonic: Alt-R

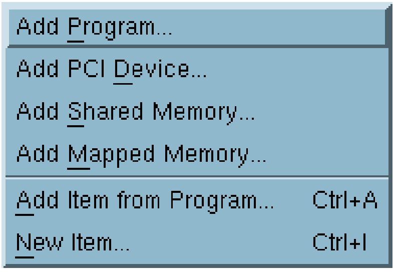


Figure 3-5. Resource menu

These menu items activate resource selection as described in “Resources” on page 3-4

Control

Mnemonic: Alt-C

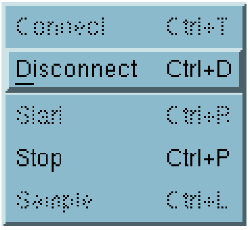


Figure 3-6. Control menu

See "Sampler Control Area" on page 3-7 for a description of these controls.

Tools

Mnemonic: Alt-L

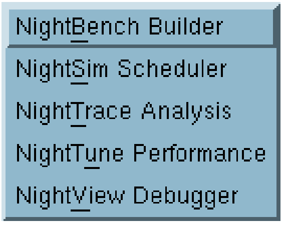


Figure 3-7. Tools menu

The following describe the options on the Tools menu:

NightBench Builder
Mnemonic: B

Opens the NightBench Program Development Environment. NightBench is a set of graphical user interface (GUI) tools for developing software with the Concurrent C/C++ and MAXAda™ compiler toolsets.

NOTE:

The Concurrent C/C++ compiler is not currently available on RedHawk systems.

See also:

- *NightBench User's Guide* (0890480)

NightSim Scheduler

Mnemonic: S

Opens the NightSim Application Scheduler. NightSim is a tool for scheduling and monitoring real-time applications which require predictable, repetitive process execution. With NightSim, application builders can control and dynamically adjust the periodic execution of multiple coordinated processes, their priorities, and their CPU assignments.

See also:

- *NightSim User's Guide* (0890480)

NightTrace Analysis

Mnemonic: T

Opens the NightTrace Analyzer. The NightTrace Analyzer is a graphical tool for analyzing the dynamic behavior of multiprocess and/or multiprocessor user applications and operating system activity. NightTrace allows you to control user and kernel trace collection daemons and can graphically display the interplay between many real-time programs and processes across multiple processors and systems.

See also:

- *NightTrace Manual* (0890398)

NightTune Performance

Mnemonic: U

Opens the NightTune Performance Analyzer. NightTune is a graphical tool for analyzing the status of the system in terms of processes, interrupts, context switches, interrupt CPU affinity, processor shielding and hyperthreading control as well as network and disk activity. NightTune can adjust the scheduling attributes of individual or groups of processes, including priority, policy, and CPU affinity.

NightView Debugger

Mnemonic: V

Opens the NightView Source-Level Debugger. NightView is a graphical source-level debugging and monitoring tool specifically designed for real-time applications. NightView can monitor, debug, and patch multiple real-time processes running on multiple processors with minimal intrusion.

See also:

- *NightView User's Guide* (0890395)

Help

Mnemonic: Alt-H

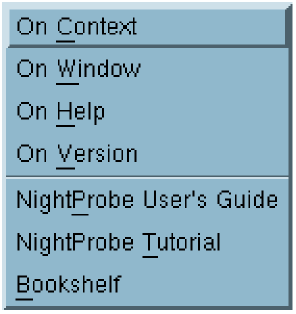


Figure 3-8. Help menu

The following describe the options on the Help menu:

On Context
Mnemonic: C

Gives context-sensitive help on the various menu options, dialogs, or other parts of the user interface.

Help for a particular item is obtained by first choosing this menu option, then clicking the mouse pointer on the object for which help is desired (the mouse pointer will become a floating question mark when the On Context menu item is selected).

In addition, context-sensitive help may be obtained for the currently highlighted option by pressing the F1 key. HyperHelp™, NightProbe's online help system, will open with the appropriate topic displayed.

On Window
Mnemonic: W

Displays help information for the current window.

On Help
Mnemonic: H

Displays help information about how to use HyperHelp, NightProbe's online help system.

On Version
Mnemonic: V

Displays a short description of the current version of NightProbe.

NightProbe User's Guide
Mnemonic: P

Opens the online version of the *NightProbe User's Guide* in the HyperHelp viewer.

NightProbe Tutorial
Mnemonic: T

Opens HyperHelp, NightProbe’s online help system, to the section containing a tutorial which shows some of the commonly used features of NightProbe.

Bookshelf
Mnemonic: B

Opens a HyperHelp window that lists all of the currently available HyperHelp publications.

4
Target System Selection Window

4
Target System Selection Window

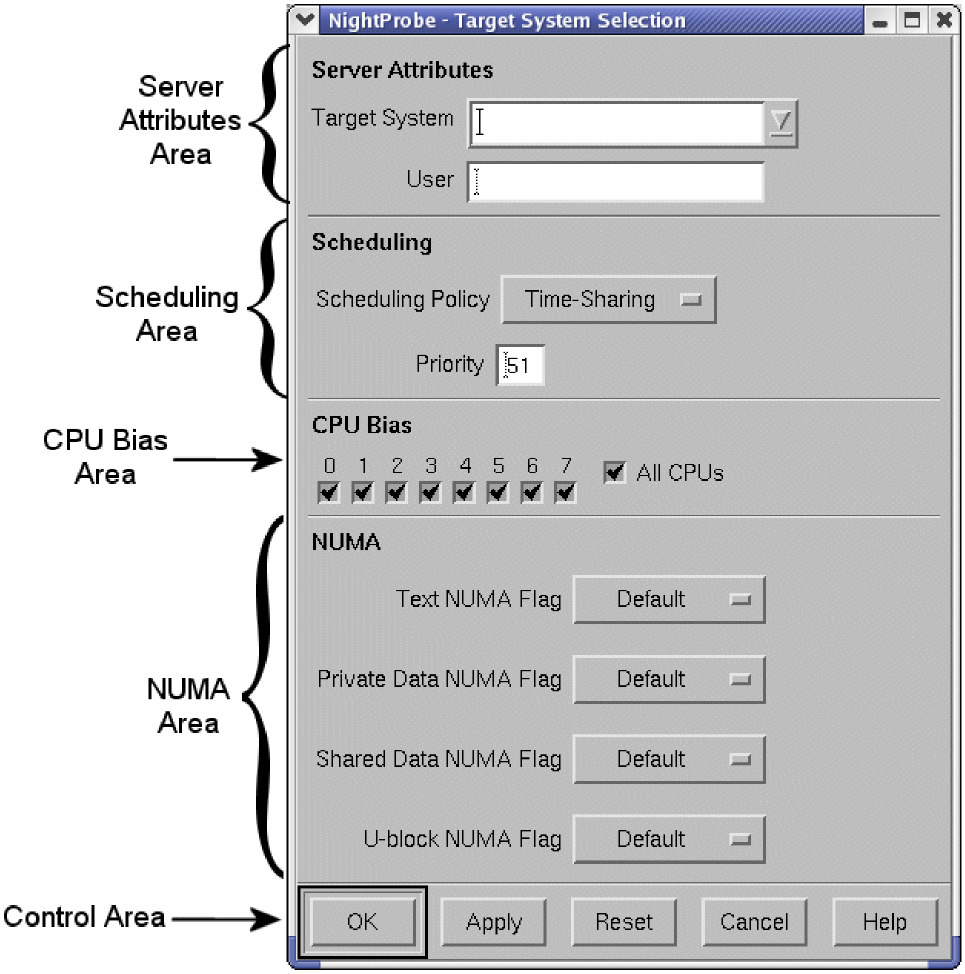
The Target System Selection window allows you to specify the system on which the target resources exist, the login name of the user connecting to the target system, and the run-time attributes associated with the data monitoring activities on the target.

It is launched from the Properties... menu option of the Target icon in the Session Overview area of the Main window.

This window is be divided into the following areas:

- Server Attributes Area (see “Server Attributes Area” on page 4-2)
- Scheduling Area (see “Scheduling Area” on page 4-3)
- CPU Bias Area (see “CPU Bias Area” on page 4-5)
- NUMA Area (see “NUMA Area” on page 4-5)
- Control Area (see “Control Area” on page 4-6)

Figure 4-1 identifies the location of each of these components in the Target System Selection window.



Server Attributes

Target System

User

Scheduling

Scheduling Policy: Time-Sharing

Priority: 51

CPU Bias

0 ☑ 1 ☑ 2 ☑ 3 ☑ 4 ☑ 5 ☑ 6 ☑ 7 ☑ ☑ All CPUs

NUMA

Text NUMA Flag: Default

Private Data NUMA Flag: Default

Shared Data NUMA Flag: Default

U-block NUMA Flag: Default

Figure 4-1. Components of the Target System Selection Window

Server Attributes Area

The server attributes area consists of the following fields:

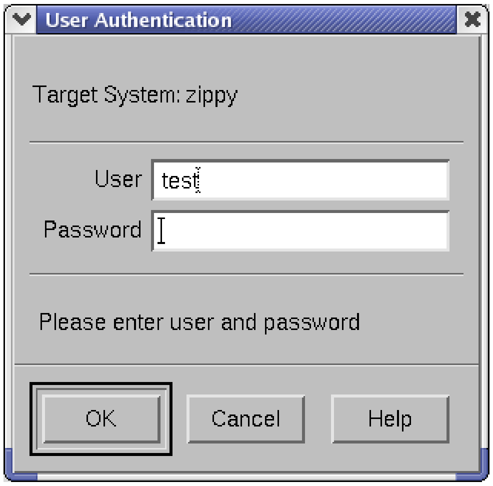
Target System — The Target System field is the name of the system to be probed.

User — The User field specifies the login name of the user connecting to the target system.

User Authentication

User authentication may be necessary when NightProbe attempts to connect to the target system as the specified user.

If user authentication is required, the following dialog will be presented when the Connect button on the Main window is pressed (see “Sampler Control Area” on page 3-7).



User Authentication

Target System: zippy

User test

Password

Please enter user and password

Figure 4-2. User Authentication Window

The User Authentication dialog contains the following fields:

User The name of the user connecting to the target system.

Password The password for the specified User on the target system.

Once authentication has succeeded, NightProbe will no longer require you to re-authenticate even if you disconnect and reconnect, unless you change the target system or user name, or exit NightProbe. Note that NightProbe does not store any password information on disk.

Scheduling Area

The Scheduling Area allows you to select a scheduling policy and assign a priority for the data monitoring activities on the target system.

Scheduling Policy

POSIX defines three types of policies that control the way a process is scheduled by the operating system. They are `SCHED_FIFO` (FiFO), `SCHED_RR` (Round Robin), and `SCHED_OTHER` (Time-Sharing). See either the *PowerMAX OS*

Programming Guide (0890423) or the *RedHawk Linux User's Guide* (0898004) for more detailed information regarding these policies and their associated classes.

FIFO

The FIFO (first–in–first–out) policy (`SCHED_FIFO`) is associated with the fixed-priority class in which critical processes and LWPs can run in predetermined sequence. Fixed priorities never change except when a user requests a change.

This policy is almost identical to the Round Robin (`SCHED_RR`) policy. The only difference is that a process scheduled under the FIFO policy does not have an associated time quantum. As a result, as long as a process scheduled under the FIFO policy is the highest priority process scheduled on a particular CPU, it will continue to execute until it voluntarily blocks.

Round Robin

The Round Robin policy (`SCHED_RR`), like the FIFO policy, is associated with the fixed-priority class in which critical processes and LWPs can run in predetermined sequence. Fixed priorities never change except when a user requests a change.

A process that is scheduled under this policy (as opposed to the FIFO policy) has an associated time quantum.

Time-Sharing

The Time-Sharing policy (`SCHED_OTHER`) is associated with the time-sharing class, changing priorities dynamically and assigning time slices of different lengths to processes in order to provide good response time to interactive processes and LWPs and good throughput to CPU-bound processes and LWPs.

Priority

The Priority is relative to the selected Scheduling Policy (see “Scheduling Policy” on page 4-3) and the range of allowable values is dependent on the operating system.

For example, on PowerMAX OS systems, the priority values for the FIFO class include 0..59, where 59 is the most urgent user priority available on the system.

On RedHawk systems, the priority values for the FIFO class include 1..99, where 99 is the most urgent user priority available on the system.

CPU Bias Area

The CPU Bias Area allows you to select certain CPUs on the target system which can be used for monitoring and recording data.

CPU Bias

Select particular CPUs by checking the desired checkboxes.

All CPUs

Selects all CPUs on the target system.

NUMA Area

On platforms belonging to the local/global/remote subclass of non-uniform memory access (NUMA) architectures, primary memory is divided into global and local memories.

Global memory is located on a memory board where it is equally distant, in terms of access time, from all of the CPUs in the system. All CPUs share a single data path to global memory known as the system bus.

Local memory is located on a CPU board where it is closer, in terms of access time, to the co-resident CPUs. The path between a CPU and its local memory does not include the system bus. Local memory usage improves the throughput of the system in two ways: smaller access times for the co-resident CPUs and less system bus contention for the remaining CPUs.

Applications can influence the page placement decisions made by the kernel by selecting NUMA policies for different parts of their address spaces. NUMA policies specify where data should reside in the local/global/remote hierarchy.

NOTE

These settings are ignored for non-NUMA target systems architectures, such as PowerHawk, PowerStack, and iHawk series machines.

Text NUMA Flag

This item selects the NUMA policy to apply to text (code) pages.

See “Policies” in the section below for a list of applicable NUMA policies.

Private Data NUMA Flag

This item selects the NUMA policy to apply to private data pages.

See “Policies” in the section below for a list of applicable NUMA policies.

Shared Data NUMA Flag

This item selects the NUMA policy to apply to shared data pages.

See “Policies” in the section below for a list of applicable NUMA policies.

U-block NUMA Flag

This item selects the NUMA policy to apply to kernel data structures that contain the stack used during system calls, the register save area used during context switches, and miscellaneous other bits of information about the LWP.

See “Policies” in the section below for a list of applicable NUMA policies.

Policies

Each of the above flags can be set to one of the following:

Global

Specifies that the designated pages should be placed in global memory.

Soft Local

Specifies that the designated pages be placed in local memory if space is available, otherwise they should be placed in global memory.

Hard Local

Specifies that the designated pages must be placed in local memory.

Default

Specifies that the default NUMA policy on the target system should be used.

Control Area

The control area is located along the bottom of the Target System Selection window and consists of the following buttons:

OK — The OK button updates the current configuration with the values specified in this dialog and closes the dialog.

Apply — The Apply button updates the current configuration with the values specified in this dialog but leaves the dialog open.

Reset — The Reset button returns all values their their initial state at the time this dialog was opened.

Close — The Close button closes the Target System Selection window without making any changes.

Help — The Help button opens the HyperHelp viewer displaying the online help topic for the Target System Selection window. See “Getting Help” on page 2-2 for details.

5
Timing Source Configuration

5
Timing Source Configuration

Timer Selection

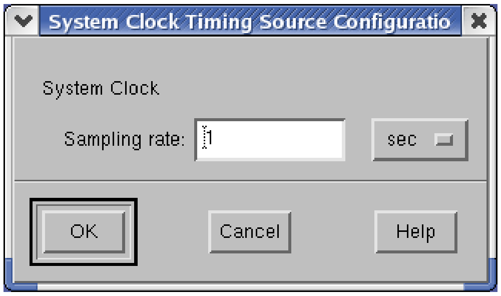
The selection of the timing source controls when and at what rate data sampling will occur.

On Demand sampling means that samples will only be taken when you press the Sample button on the Main window.

This chapter discusses the dialogs relating to the selection of iterative timing sources: System Clock and Frequency Based Scheduling.

System Clock Timing Source Configuration

Selecting the System Clock... menu option from the Timer icon in the Session Overview area of the Main window launches the System Timing Source Configuration window:



System Clock Timing Source Configuratio

System Clock

Sampling rate: 1 sec

Figure 5-1. System Clock Timing Source Configuration dialog

Selection of this timing source means the sampler will use the system clock as the timing source at the frequency you select in the dialog. Choose a unit of time measurement using the option menu to the right of the Sampling rate text field and then enter the amount of time that should pass between samples. The interval you select is displayed in to the right of the Timer icon in the Session Overview area.

You can change the rate subsequently by selecting the Properties... option from the option menu on the Timer icon in the Session Overview area of the Main window.

FBS Timing Source Configuration

Selecting the Frequency Based Scheduling... menu option from the Timer icon menu in the Session Overview area of the Main window launches the FBS Timing Source Configuration window:

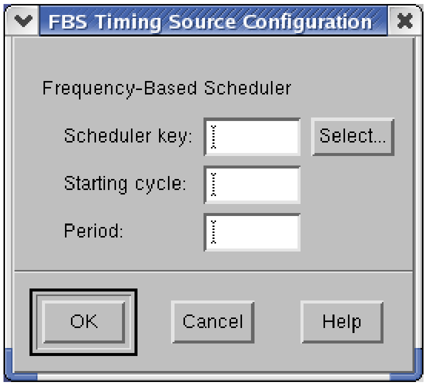


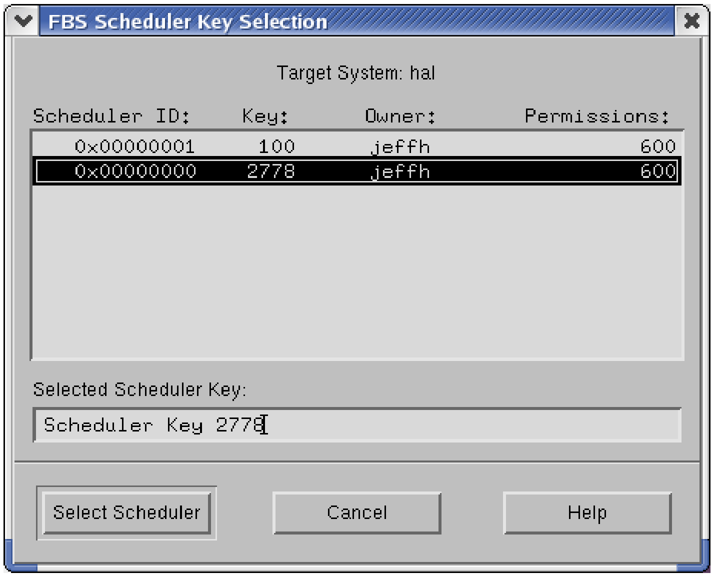
Figure 5-2. Frequency-Based Scheduler Configuration dialog

Selecting this option means that the NightProbe will take samples as directed by a frequency-based scheduler. This allows for synchronization of data sampling with the application.

Use this dialog to configure the sampling interval. You must specify the Scheduler Key for a configured scheduler, and specify the Starting cycle within each frame and Period (cycle interval) where you want samples taken. The frequency-based scheduler that you specify must be running by the time you connect NightProbe.

To select a scheduler that has already been set up and configured, press the Select... button to display the Frequency-Based Scheduler Selection window:

FBS Selection



Target System: hal

Scheduler ID:	Key:	Owner:	Permissions:
0x00000001	100	jeffh	600
0x00000000	2778	jeffh	600

Selected Scheduler Key:

Scheduler Key 2778

Figure 5-3. Frequency-Based Scheduler Selection dialog

Selecting the desired scheduler from the list populates the Selected Scheduler Key field appropriately and the choice will be propagated to the Scheduler Key field of the Frequency-Based Scheduler Configuration dialog when you press either Apply or OK.

Changing the Timing Selection Properties

Selecting the Properties... menu option from the Timer icon menu in the Session Overview area of the Main window will launch the appropriate timing source dialog so that you can make adjustments.

Double-clicking the Timer icon has the same effect.

6
Output Configuration

6
Output Configuration

Output Selection

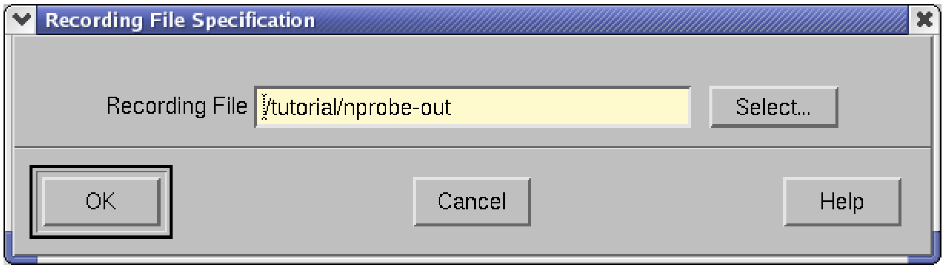
The selection of the output methods controls where sampled data is sent or viewed.

There are five output methods available:

- File output
- NightTrace output
- List Window output
- Spreadsheet output
- Program output

File Specification

The File Specification window is launched from the File Output... menu option of the Outputs icon menu in the Session Overview area of the Main window.



Recording File /tutorial/nprobe-out

Figure 6-1. The Recording File Specification Window

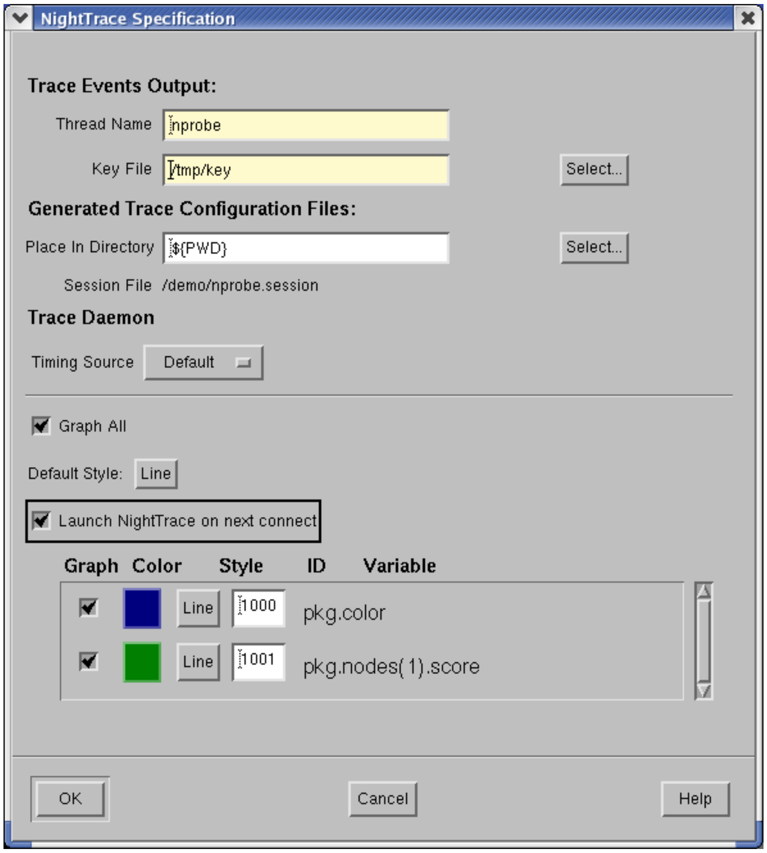
This output option allows you to specify a file as a destination for the output of the sampler (the recording file).

Figure 6-1 shows the Recording File Specification window that appears when you select the To File option. You may specify the file in one of two ways: by typing in the pathname in the Recording File field or by clicking on the Select button and using the File Selection window that appears to select a new or existing file.

The data recording output file can subsequently be processed using the NightProbe API (see Chapter 15 "NightProbe API") or can be viewed using the List Window dialog from within NightProbe (see "List Window" on page 6-6).

NightTrace Specification

The NightTrace Specification window is launched from the NightTrace Output... menu option of the Outputs icon menu in the Session Overview area of the Main window.



Trace Events Output:

Thread Name nprobe

Key File /tmp/key

Generated Trace Configuration Files:

Place In Directory ${PWD}

Session File /demo/nprobe.session

Trace Daemon

Timing Source Default

☑ Graph All

Default Style: Line

☑ Launch NightTrace on next connect

Graph	Color	Style	ID	Variable
☑	🟦	Line	1000	pkg.color
☑	🟩	Line	1001	pkg.nodes(1).score

Figure 6-2. The NightTrace Specification Window

This output option allows you to save the sampled data in the form of NightTrace records that can be streamed to a NightTrace daemon for collection.

Each sampled data value will be logged as a trace event that can be viewed using NightTrace. In addition, the value of those data values can be graphed on a NightTrace Data Graph.

See the *NightTrace User's Guide* for more information (0890398).

Trace Events Output

Thread Name

Trace events in NightTrace are associated with a particular thread. Trace events associated with the data streamed from this NightProbe session will have the value of this field as their thread name in NightTrace. This can be useful in cases where data is streamed to NightTrace from multiple data sources; the thread name can help to determine the data source.

The default Thread Name for data being streamed to NightTrace from NightProbe is `nprobe`.

Key File

The Key File helps to synchronize the transfer of data from NightProbe to NightTrace. When the NightTrace daemon is started, it is given the value of this key file so that it can receive the trace event data from NightProbe. For instance, if the key file was **/tmp/mykeyfile**, the NightTrace daemon could be invoked with the key file:

```
ntraceud /tmp/mykeyfile
```

Select...

Presents a file selection dialog from which to select the key file.

Generated Trace Configuration Files

Place In Directory

NightProbe generates support files that will be used by NightTrace. This directory specifies where NightProbe should save them.

The default location is the current working directory.

Select...

Presents a file selection dialog from which to select the directory in which to store the generated NightTrace configuration files.

Session File

Session configuration files contain information (such as display page configurations and daemon definitions) specific to a particular session of NightTrace. Information related to this session of NightProbe will be stored in the file listed here.

Graph All

When this checkbox is selected, Data Graphs will be created in the generated NightTrace display page for all monitored variables. You may make individual selections via the Graph checkbox associated with each variable in the list at the bottom of this dialog (see "Graph" on page 6-4).

Default Style

Values in the Data Graphs on the NightTrace display page associated with this session of NightProbe may appear as either lines or bars of varying height. The height of the bar or line reflects the value of the variable.

The choice of Default Style determines the default style for all monitored variables. You may make individual selections via the Style option associated with each variable in the list at the bottom of this dialog (see "Style" on page 6-5).

Launch NightTrace on next connect

When this option is selected, NightTrace will be started the next time that NightProbe connects to the target application (see "Sampler Control Area" on page 3-7).

You may then start the daemon either through the NightTrace GUI or by invoking the user daemon **`ntraceud(1)`** from the command line.

If NightProbe disconnects from the target application and changes are made in this dialog, NightTrace must reload its configuration files to be informed of the changes.

NOTE

> If this option is not selected, NightTrace may be started by issuing **`ntrace(1)`** on the command line or by selecting the NightTrace Analysis menu item from the Tools menu on the Main window (see "Tools" on page 3-12). See the *NightTrace Manual* (0890398) for more information on this tool.

The bottom portion of this dialog contains information specific to each of the monitored variables.

Graph

When this checkbox is selected, a Data Graph will be created in the generated NightTrace display page for the associated variable. You may select all variables via the Graph All checkbox that appears in the upper portion of this dialog (see "Graph All" on page 6-3).

Color

Specifies the color of the bar or line representing the trace event associated with the variable.

Clicking on the color presents a color selection dialog.

Style

Trace events may appear as either lines or bars of varying height in the Data Graphs on the NightTrace display page associated with this session of NightProbe. The height of the bar or line reflects the value of the variable.

The value selected here determines the style of the associated variable. The default style is determined by the Default Style setting that appears in the upper portion of this dialog (see “Default Style” on page 6-4).

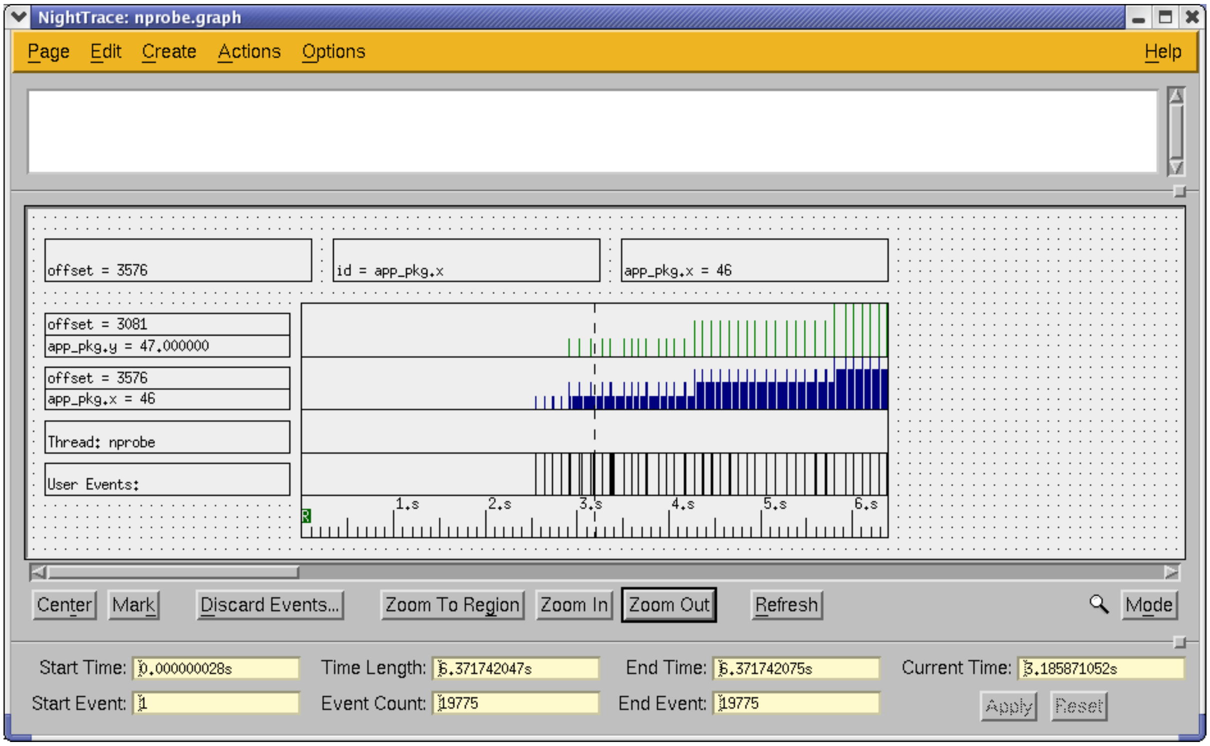
ID

Unique identifier used as the trace event ID for the trace events associated with this variable.

Variable

The name of the monitored variable.

Figure 6-2 shows a NightTrace display page generated using the To NightTrace output method. Data Graphs for two variables appear on the grid. The Data Graph associated with the variable `app_pkg.x` contains blue bars of varying heights; the Data Graph associated with the variable `app_pkg.y` consists of green lines of varying heights.



offset = 3576

id = app_pkg.x

app_pkg.x = 46

offset = 3081
app_pkg.y = 47.000000

offset = 3576
app_pkg.x = 46

Thread: nprobe

User Events:

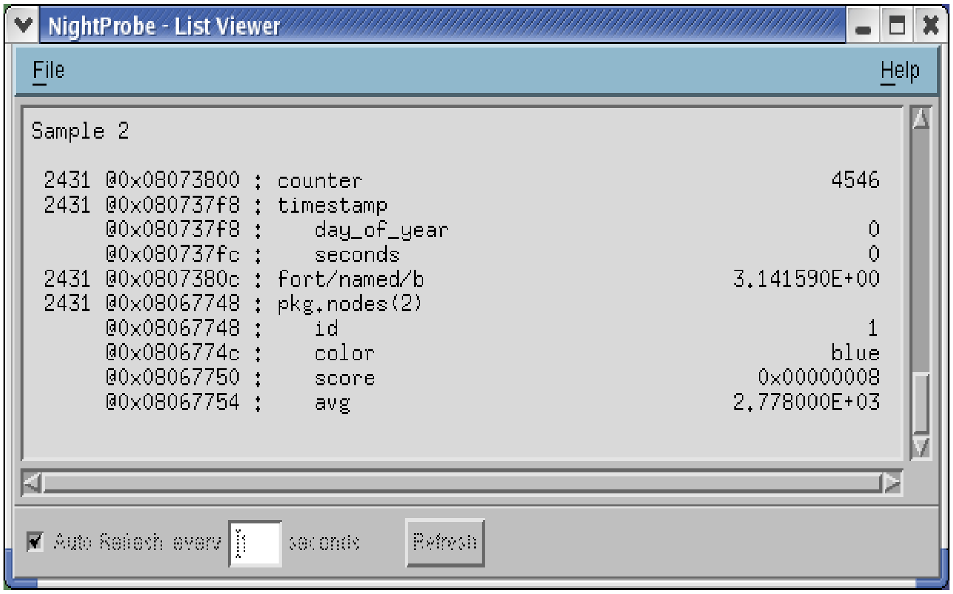
Start Time: 0.000000028s
Time Length: 6.371742047s
End Time: 6.371742075s
Current Time: 3.185871052s

Start Event: 1
Event Count: 19775
End Event: 19775

Figure 6-3. NightTrace display page

List Window

The List Viewer window is launched from the List Window Output menu option of the Outputs menu in the Session Overview area of the Main window.



```
Sample 2

 2431 @0x08073800 : counter                                    4546
 2431 @0x080737f8 : timestamp
      @0x080737f8 :    day_of_year                                0
      @0x080737fc :    seconds                                    0
 2431 @0x0807380c : fort/named/b                       3.141590E+00
 2431 @0x08067748 : pkg.nodes(2)
      @0x08067748 :    id                                         1
      @0x0806774c :    color                                   blue
      @0x08067750 :    score                             0x00000008
      @0x08067754 :    avg                             2.778000E+03
```

☑ Auto Refresh every 1 seconds

Figure 6-4. The List Viewer Window

The List Viewer window, shown in Figure 6-4, is the simpler of the two viewing windows. It allows you to:

- View a scrolled text report on the sampled data.
- View previously recorded data files as text within a scrolled window.
- Display sampled data after every sample, after a set number of samples, or upon demand.

The List Viewer window contains

- The menu bar (see “Menu Bar” on page 6-7)
- The scrolled text Viewing Area
- The Control Area (see “Control Area” on page 6-8)

Menu Bar

The List Viewer window menu bar contains File and Help menus. These are described in the next two sections.

File

Mnemonic: F

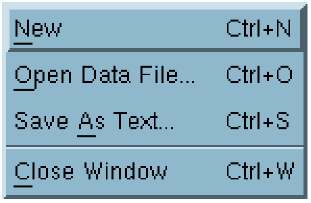


Figure 6-5. File menu

New
Mnemonic: N
Accelerator: <Control><N>

> This option allows you to clear the scrolled text viewing area. If you are monitoring a running program, you will not be able to recall the erased information in this window.

Open Data File...
Mnemonic: O
Accelerator: <Control><O>

> This option allows you to open a data file that was created using the To File option of the Output menu. The data file will be translated to ASCII text and displayed in the scrolled text viewing area.

Save As...
Mnemonic: A
Accelerator: <Control><S>

> This option allows you to save the current contents of the text area (including what is not visible in the viewing area) to a file. You will be presented with a file selection dialog with which to choose a file name.

Close Window
Mnemonic: C
Accelerator: <Control><W>

> Using this option closes this window and removes it from the Outputs list.

Help

Mnemonic: H

The Help menu operates exactly like the menu provided in the Main window. It lists a number of topics on which help is available, and selecting any topic will display a help window. See "Getting Help" on page 2-2 for details.

Control Area

The control area appears at the bottom of the List Viewer window. It allows you to control when new information is added to the viewing area.

Auto Refresh

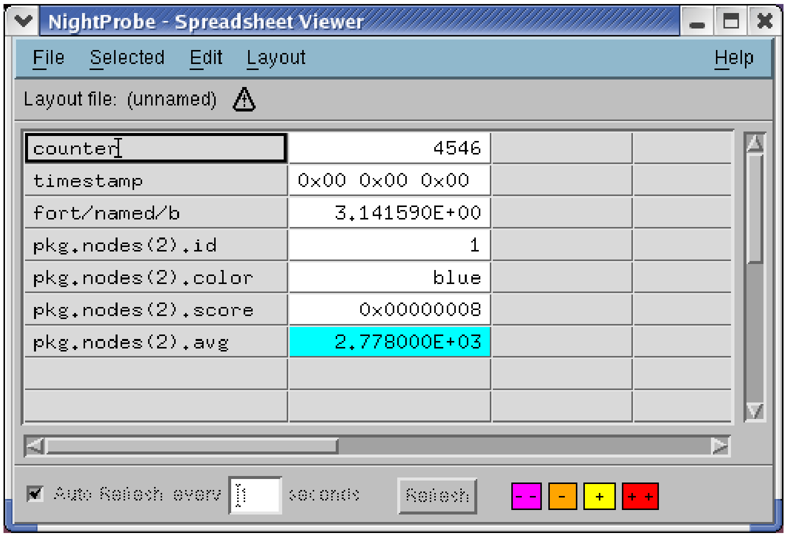
The Auto Refresh checkbox and text entry field control how often the sampled values are displayed. The List Viewer is designed for displaying values at human-readable rates, not necessarily displaying all sampled values (especially if the sampling rate is extremely fast).

Refresh

The Refresh button can be used when Auto Refresh is turned off. The Refresh button gets the most recent sample taken and displays it in the List Viewer window. Note that the Refresh button does not cause the sampler to take a new sample or record a sample to a file.

Spreadsheet

The Spreadsheet Viewer window is launched from the Output menu on the Main window.



Layout file: (unnamed) ⚠

counter	4546		
timestamp	0x00 0x00 0x00		
fort/named/b	3.141590E+00		
pkg.nodes(2).id	1		
pkg.nodes(2).color	blue		
pkg.nodes(2).score	0x00000008		
pkg.nodes(2).avg	2.778000E+03		

☑ Auto Refresh every 1 seconds

Figure 6-6. The Spreadsheet Viewer Window

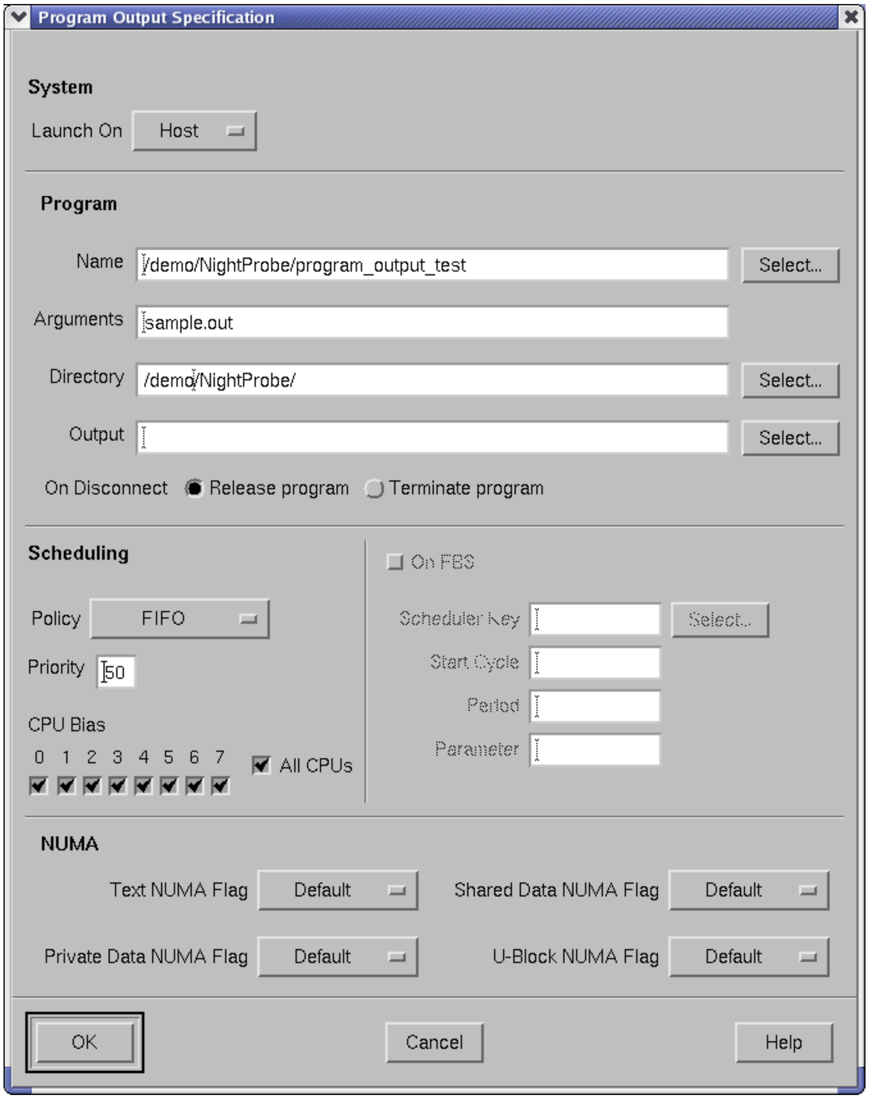
The Spreadsheet Viewer as shown in Figure 6-6, allows you to monitor and modify variables while the program is running. The Spreadsheet Viewer window has many configuration options and is described in detail in its own section, "Spreadsheet Viewer" on page 14-1.

To Program

Mnemonic: P

The To Program option on the Output menu presents you with the Program Output Specification window as shown in Figure 6-7.

This dialog allows you to specify a program that will be used to process the data recording output using the NightProbe Application Programming Interface (see Chapter 15 "NightProbe API").



Program Output Specification

System

Launch On: Host

Program

Name: /demo/NightProbe/program_output_test

Arguments: sample.out

Directory: /demo/NightProbe/

Output:

On Disconnect ● Release program ○ Terminate program

Scheduling

Policy: FIFO

Priority: 50

CPU Bias

0	1	2	3	4	5	6	7
☑	☑	☑	☑	☑	☑	☑	☑

☑ All CPUs

□ On FBS

Scheduler Key:

Start Cycle:

Period:

Parameter:

NUMA

Text NUMA Flag: Default

Shared Data NUMA Flag: Default

Private Data NUMA Flag: Default

U-Block NUMA Flag: Default

Figure 6-7. The Program Output Specification Window

System

Launch On

Indicates whether the program should be executed on the Host system (where the GUI portion of NightProbe is running) or the Target system (where the probed application is running).

Program

Name

The name of the program used to process the data recording output streamed from NightProbe.

Select...

Presents a file selection dialog from which to select the desired program.

NOTE

Pathnames are relative to the host system.

Arguments

Any arguments to be passed to the program are entered in this field.

Directory

The current working directory for the program.

Select...

Presents a file selection dialog from which to select the desired directory.

NOTE

Pathnames are relative to the host system.

On Disconnect

This selection determines how NightProbe handles the program which processed the data recording output when NightProbe disconnects from the target program.

Release Program

Allow the program which processed the data streamed from NightProbe to continue running after NightProbe has disconnected from the target program (see “Sampler Control Area” on page 3-7).

Terminate Program

Terminates the program which processed the data streamed from NightProbe to continue running after NightProbe has disconnected from the target program (see “Sampler Control Area” on page 3-7).

Scheduling

Policy

This panel of radio buttons allows you to select the POSIX scheduling policy for the specified program. The options are as follows: the first–in–first–out scheduling policy (FIFO), the round–robin scheduling policy (Round Robin), and the other scheduling policy (Time-Sharing)

For a full description of the behavior of processes that are scheduled under the respective policies on PowerMAX OS systems, refer to the "Process Scheduling and Management" chapter of the *PowerMAX OS Programming Guide* (0890423); on RedHawk Linux systems, refer to the "Process Scheduling" chapter of the *RedHawk Linux User's Guide* (0898004).

Priority

The priority of the program used to process data recording output. The range of priority values that you can enter is governed by the scheduling policy specified (see "Policy" above). Higher numerical values correspond to more favorable scheduling priorities.

For complete information on scheduling policies and priorities on PowerMAX OS systems, refer to the "Process Scheduling and Management" chapter of the *PowerMAX OS Programming Guide* (0890423); on RedHawk Linux systems, refer to the "Process Scheduling" chapter of the *RedHawk Linux User's Guide* (0898004).

CPU Bias

This panel of checkbuttons allows you to select the processor or processors on which the program processing data recording output can be scheduled.

You can choose to run the program that processes the data recording output on a different CPU from the CPU on which their shielded application is running, thereby reducing interference.

All CPUs

Specifies that the program processing data recording output can be scheduled on all available processors.

On FBS

This checkbox should be checked if the program processing the data recording output is to be scheduled on the frequency-based scheduler.

Scheduling the program in cycles unused by the probed application allows for minimal interference with that application.

Scheduler Key

This field allows you to specify the *key* of the frequency–based scheduler that you wish to use. The key is a user–chosen numeric identifier with which the scheduler is associated.

Select...

Presents the Frequency-Based Scheduler Selection dialog (see "FBS Timing Source Configuration" on page 5-2) allowing you to select a frequency-based scheduler on the target system.

Start Cycle

This field allows you to specify the first *minor cycle* in which the specified program is to be wakened in each *major frame*. Enter a number ranging from zero to the total number of minor cycles per frame minus one.

Selecting a start cycle which is not used by the probed application will reduce the interference with that application.

Period

This field allows you to establish the frequency with which the specified program is to be wakened in each *major frame*. A period of one indicates that the specified program is to be wakened every *minor cycle*; a period of two indicates that it is to be wakened once every two minor cycles, etc. Enter the number of minor cycles representing the frequency with which you wish the program to be wakened. This number can range from zero to the number of minor cycles that compose a frame on the scheduler.

Parameter

This field allows you to pass an integer value to the program when it is scheduled on the frequency–based scheduler. The value must be a 32-bit decimal value. The default is no value.

NUMA

NOTE

The NUMA flags only apply to programs running on a PowerMAX OS system.

Text NUMA Flag

This item selects the NUMA policy to apply to text (code) pages.

See "Policies" on page 4-6 for a list of applicable NUMA policies.

Private Data NUMA Flag

This item selects the NUMA policy to apply to private data pages.

See "Policies" on page 4-6 for a list of applicable NUMA policies.

Shared Data NUMA Flag

This item selects the NUMA policy to apply to shared data pages.

See "Policies" on page 4-6 for a list of applicable NUMA policies.

U-Block NUMA Flag

This item selects the NUMA policy to apply to kernel data structures that contain the stack used during system calls, the register save area used during context switches, and miscellaneous other bits of information about the LWP.

See "Policies" on page 4-6 for a list of applicable NUMA policies.

7
Program Window

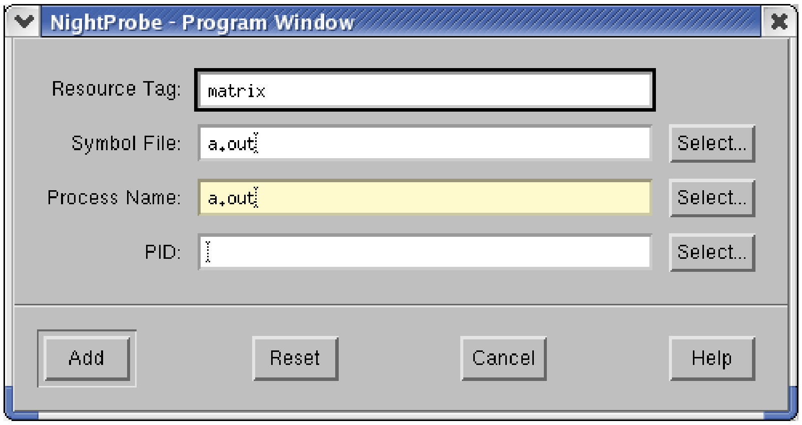
7
Program Window

The Program window specifies a program to be probed.

It is launched from the Add Program... menu option of the Resources icon in the Main window.

This window provides for the selection of four pieces of information as shown in Figure 7-1:

- Resource Tag
- SymbolFile
- Process Name
- PID



Resource Tag: matrix

Symbol File: a.out

Process Name: a.out

PID:

Figure 7-1. The Program Window

Resource Tag

A Resource Tag is just a short-hand identifier for the resource. It is important as it is used to maintain relationships between variables and resources in the session configuration file so that configurations may be saved and reloaded in the future.

By default, a unique resource tag is assigned when the window is launched.

If you do not modify the default value, it will be automatically changed to the basename of the value used in the Process Name field when the Add button is pressed.

If the tag name is modified, it will only be adjusted if it conflicts with another resource in which case NightProbe will append an underscore and number to the value of the Resource Tag field when the Add button is pressed and pop up a dialog indicating the conflict and tag adjustment.

Symbol File

A Symbol File is an executable file which contains symbolic and debug information that allows NightProbe to identify and list variables that can be probed.

A Symbol File is not strictly required for probing a process, but without it, you can only define artificial variables using the Item Definition dialog.

If no Symbol File is specified, but a filename is specified for the Process Name, NightProbe will automatically attempt to use that filename as the symbol file.

If you specify a file name for the Symbol File and have not specified a value for the Process Name, NightProbe automatically fills in the Process Name with the specified file name.

If specified, the Symbol File must exist, be accessible from the host system, and be a valid executable program file.

Process Name

A Process Name is required in order to probe programs. The Process Name is used to identify running processes when no PID is specified, or when the specified PID cannot be found.

When attempting to connect, if the specified PID is not found, NightProbe will attempt to locate a process that matches the Process Name. If a matching process can be located, NightProbe pops up a dialog indicating the situation and asks if the connection should proceed or be terminated.

Normally, the Process Name and Symbol File are the same but they do not have to be. The Process Name could be a file name of a stripped executable. The Process Name file does not have to exist on the host system.

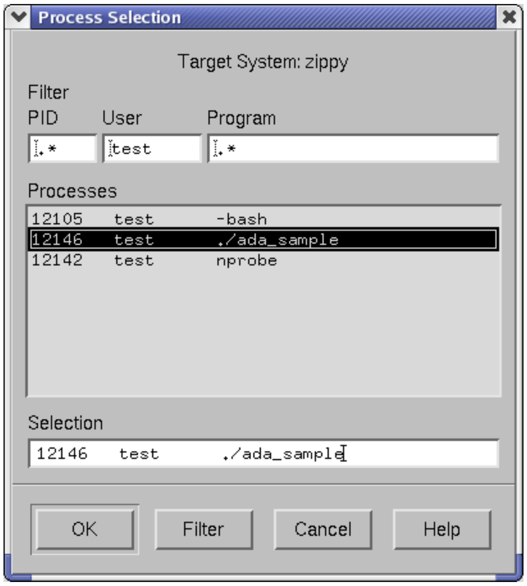
The Process Name is the only required piece of information that must be entered in this dialog. It is normally sufficient to just type in the name of the executable file the Process Name text field or select it using the Select... file dialog and then click Add to exit the dialog.

PID

The PID is an optional field which specifies a specific process ID to probe.

If the PID is left blank or if the specified PID is not running when you Connect, NightProbe will automatically attempt to locate a running process that matches the Process Name. If the PID was blank, NightProbe silently proceeds to connect using the process ID it located. If the PID was not blank, NightProbe will pop up a dialog indicating the newly located PID and ask whether the connection should proceed or be terminated.

The Select... button brings up the Process Selection window, which presents a list of the process IDs, owner user names, and process names running on the target system and allows you to select one for monitoring. Selecting a file in the Process Selection window automatically fills in the Process Name, Resource Tag, and PID in the Program window.



PID: .*
User: test
Program: .*

```
12105   test       -bash
12146   test       ./ada_sample
12142   test       nprobe
```

Selection: 12146   test   ./ada_sample

Figure 7-2. The Process Selection Window

Filters are patterns constructed using standard regular expression syntax. The default PID and Program filter is:

```
.*
```

which means that everything is displayed. The default User filter is your login name.

When you use the Process Selection window to select a program, NightProbe attempts to construct a complete pathname based on the simple name in the process table and the current `PATH` environment variable. If a complete pathname cannot be determined, then the simple name is used and you may need to add the correct pathname to the

Process Name field in the Program window. Otherwise, NightProbe will not be able to locate the target process and will generate an error message.

8
PCI Device Window

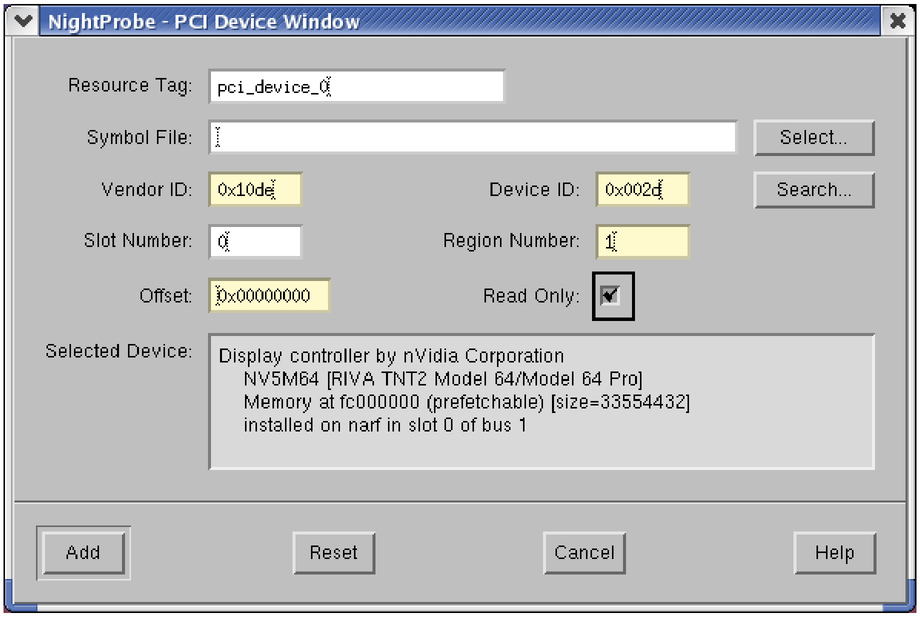
8
PCI Device Window

The PCI Device window specifies a PCI device to be monitored or recorded.

It is launched from the Add PCI Device... menu option of the Resources icon in the Main window.

This window provides for the selection of five key pieces of information as illustrated in Figure 8-1:

- Resource Tag
- Symbol File
- PCI Device Selection
- Offset
- Options



Resource Tag: pci_device_0

Symbol File:

Vendor ID: 0x10de

Device ID: 0x002d

Slot Number: 0

Region Number: 1

Offset: 0x00000000

Read Only: ☑

Selected Device: Display controller by nVidia Corporation
NV5M64 [RIVA TNT2 Model 64/Model 64 Pro]
Memory at fc000000 (prefetchable) [size=33554432]
installed on narf in slot 0 of bus 1

Figure 8-1. The PCI Device WIndow

Applicability

The PCI Device window is only available for target systems running RedHawk Linux. PCI device mapping depends on the Base Address Register file system extension to **/proc/bus/pci**, which is only available under RedHawk Linux. (**bar_scan_open(3)**)

However, PowerMAX OS users can map VME devices using the Mapped Memory window by specifying /dev/mem as the device and speciying the appropriate physical address in the Base Offset field.

Resource Tag

A Resource Tag is just a short-hand identifier for the resource. It is important as it is used to maintain relationships between variables and resources in the session configuration file so that configurations may be saved and reloaded in the future.

By default, a unique resource tag is assigned when the window is launched.

If the tag name is modified, it will only be adjusted if it conflicts with another resource in which case NightProbe will append an underscore and number to the value of the Resource Tag field when the Add button is pressed and pop up a dialog indicating the conflict and tag adjustment..

Symbol File

A Symbol File is an executable file which contains symbolic and debug information that allows NightProbe to identify and list types that can be associated with artificial variables you associate with the PCI device.

A Symbol File is not required for probing a PCI device, but without it, you can only define artificial variables using basic numeric and character types in the Item Definition dialog.

If specified, the Symbol File must exist, be accessible from the host system, and be a valid executable program file.

PCI Device Specification

PCI devices are probed by mapping regions of PCI memory into the address space of the sampler process.

Since the memory addresses of devices on the PCI bus are dynamic in nature, the specification of the device itself is the preferred approach.

Alternatively, if the address is known, you can use the Memory Mapped dialog using **/dev/mem** as the device.

NightProbe can only probe PCI devices with mappable register or memory regions. It cannot probe I/O ports.

Region numbers start at zero for each PCI device and are monotonically increasing.

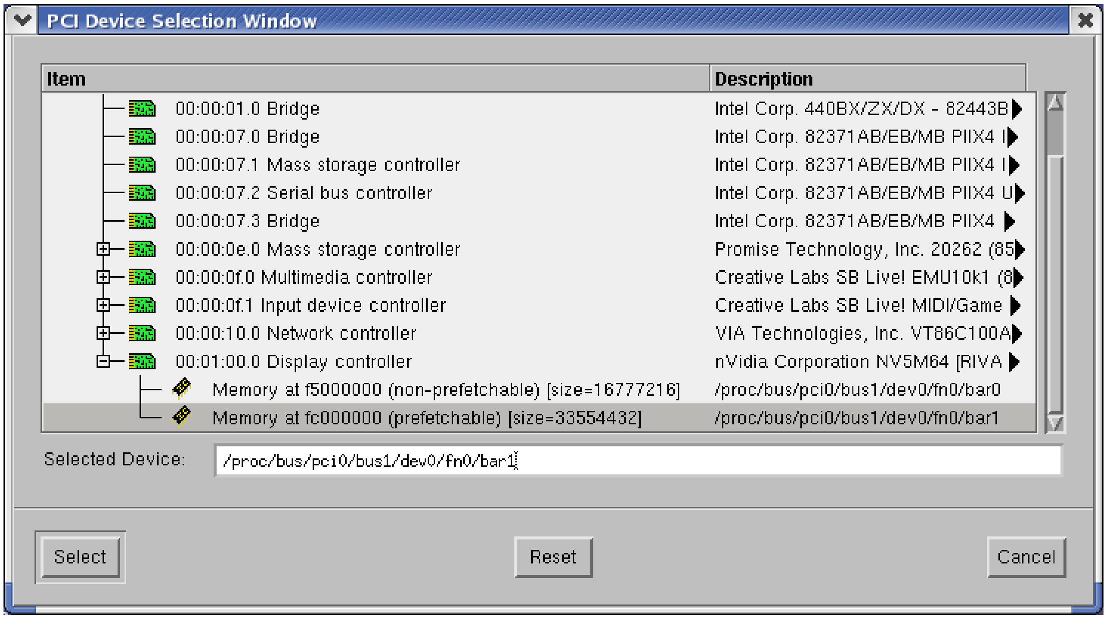
To specify a PCI device, the following pieces of information are required: Vendor ID, Device ID, Region Number, and Slot Number.

These fields only accept numeric input and are limited to values in the range 0x0 .. 0xffff. A Region Number of zero corresponds to the first mappable region for the device.

The Slot Number is not always required. It is required only to differentiate between two or more devices having identical Vendor and Device IDs installed on the same system. If it is left blank, when connecting and their are multiple such devices, NightProbe will allow you to select the appropriate device using the PCI Device Selection window. If you choose not to select a device, NightProbe will issue a diagnostic and terminate the connection.

If these values are readily available, you can enter them in the text fields.

Alternatively, pressing the Search... button will launch the PCI Device Selection window:



Item	Description
00:00:01.0 Bridge	Intel Corp. 440BX/ZX/DX - 82443B
00:00:07.0 Bridge	Intel Corp. 82371AB/EB/MB PIIX4 I
00:00:07.1 Mass storage controller	Intel Corp. 82371AB/EB/MB PIIX4 I
00:00:07.2 Serial bus controller	Intel Corp. 82371AB/EB/MB PIIX4 U
00:00:07.3 Bridge	Intel Corp. 82371AB/EB/MB PIIX4
00:00:0e.0 Mass storage controller	Promise Technology, Inc. 20262 (85
00:00:0f.0 Multimedia controller	Creative Labs SB Live! EMU10k1 (8
00:00:0f.1 Input device controller	Creative Labs SB Live! MIDI/Game
00:00:10.0 Network controller	VIA Technologies, Inc. VT86C100A
00:01:00.0 Display controller	nVidia Corporation NV5M64 [RIVA
Memory at f5000000 (non-prefetchable) [size=16777216]	/proc/bus/pci0/bus1/dev0/fn0/bar0
Memory at fc000000 (prefetchable) [size=33554432]	/proc/bus/pci0/bus1/dev0/fn0/bar1

Selected Device: /proc/bus/pci0/bus1/dev0/fn0/bar1

Figure 8-2. The PCI Device Selection Window

The PCI Device Selection window presents an interactive PCI device browser containing a list of PCI devices and their mappable regions organized in a tree.

The tree contains a list of all PCI devices installed on the target system.

PCI devices having mappable registers, or regions, are shown with an expandable control box to the left of their icon. Click the control box to see the mappable regions for each device.

The Select button will remain desensitized until you select a mappable region node. Since devices without mappable regions are not able to be probed by NightProbe, the Select button will remain desensitized if you select a PCI device node.

If the Vendor ID, Device ID, or Slot Number text fields of the PCI Device window contain data when the Search... button is pressed, a message dialog asks whether you wish to filter the list of PCI devices by the values of those fields or would prefer to see the complete list of PCI devices on the target system.

If you select the filtering option, but no PCI devices match the supplied values, a diagnostic will be issued and the entire list of PCI devices will be presented in the tree.

Offset

By default, the Offset value field is set to zero. This value defines the base offset from the beginning of the selected PCI device memory region for all artificial variables to be associated with the region. The Offset in the Item Definition dialog is added to the base offset defined here.

Options

Optionally, the Read Only checkbox may be activated. If selected, the mapping to the PCI device's memory region will be done in read-only mode to prevent accidental modification when using the Spreadsheet Viewer.

9
Shared Memory Window

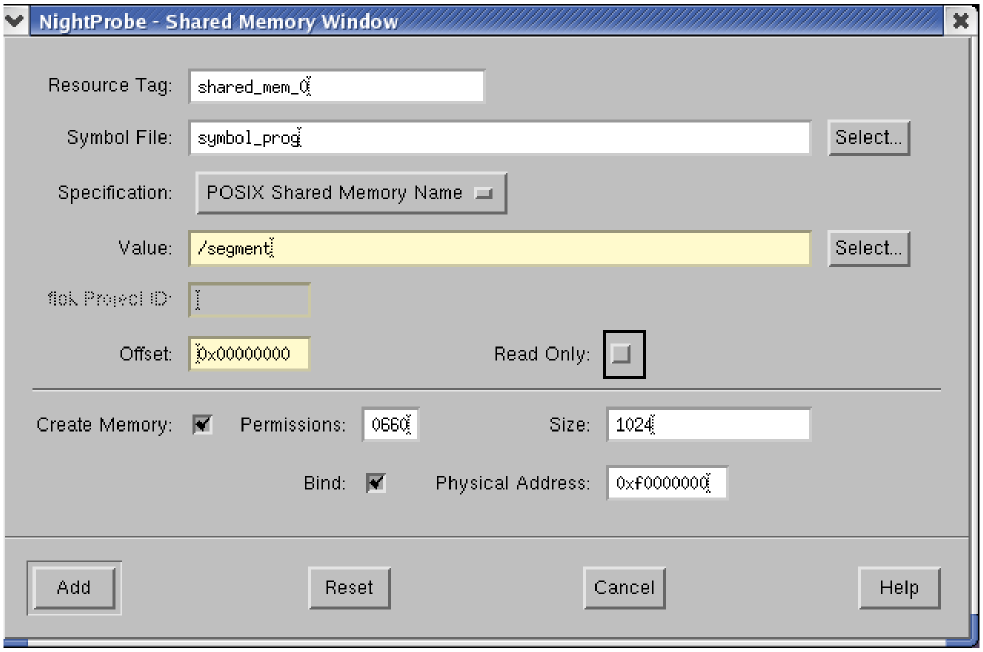
9
Shared Memory Window

The Shared Memory window specifies a shared memory segment to be monitored or recorded.

It is launched from the Add Shared Memory... menu option of the Resources icon in the Main window.

This window provides for the selection of four key pieces of information as illustrated in Figure 9-1:

- Resource Tag
- Symbol File
- Share Memory Segment Identification
- Options



NightProbe - Shared Memory Window

Resource Tag: shared_mem_0

Symbol File: symbol_prog

Specification: POSIX Shared Memory Name

Value: /segment

ftok Project ID:

Offset: 0x00000000 Read Only: □

Create Memory: ☑ Permissions: 0660 Size: 1024

Bind: ☑ Physical Address: 0xf0000000

Figure 9-1. The Shared Memory WIndow

Resource Tag

A Resource Tag is just a short-hand identifier for the resource. It is important as it is used to maintain relationships between variables and resources in the session configuration file so that configurations may be saved and reloaded in the future.

By default, a unique resource tag is assigned when the window is launched.

If the tag name is modified, it will only be adjusted if it conflicts with another resource in which case NightProbe will append an underscore and number to the value of the Resource Tag field when the Add button is pressed and pop up a dialog indicating the conflict and tag adjustment..

Symbol File

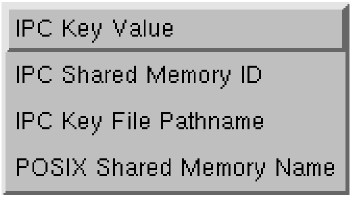
A Symbol File is an executable file which contains symbolic and debug information that allows NightProbe to identify and list types that can be associated with artificial variables you create and associate with the shared memory segment.

A Symbol File is not required for probing a shared memory segment, but without it, you can only define artificial variables using basic numeric and character types in the Item Definition dialog.

If specified, the Symbol File must exist, be accessible from the host system, and be a valid executable program file.

Shared Memory Segment Specification

The Specification field contains a menu list describing the method for identifying the shared memory segment:



IPC Key value

When this menu option is selected, the value supplied in the Value text field will be interpreted as a shared memory key as supplied to the **shmget(2)** system call.

Specification of an IPC_PRIVATE key value is inappropriate and is disallowed. The Select... button to the right of the Value text field provides a list of all existing shared memory segments on the target system.

IPC Shared Memory ID

When this menu option is selected, the value supplied in the Value text field will be interpreted as a shared memory identifier as returned from the **`shmget(2)`** system call. The Select... button to the right of the Value text field provides a list of all existing shared memory segments on the target system.

IPC Key File Pathname

When this menu option is selected, the value supplied in the Value text field will be interpreted as a pathname. The shared memory segment will be identified using the **`ftok(3X)`** service using the specified pathname and ftok Project ID value in the text field below the Value text field. The Select... button to the right of the Value text field provides a standard file selection dialog to locate specify the pathname.

Note:

The filenames provided in the file selection dialog are relative to the host system, even though the selected file must be accessible from the target system when NightProbe connects to the target system.

POSIX Shared Memory Name

When this menu option is selected, the value supplied in the Value text field will be interpreted as POSIX shared memory object as created by the **`shm_open(3)`** service. The Select... button to the right of the Value text field provides a list of all existing named POSIX shared memory objects.

The Offset value defaults to zero. This value defines the base offset from the beginning of the shared memory segment for all variables to be associated with this shared memory segment. The Offset in the Item Definition dialog is added to the base offset defined here.

Options

Optionally, the Read Only checkbox may be activated. If activated, the attachment of the shared memory region to the sampler process's address space will be done in read-only mode, preventing modification of the region via the Spreadsheet Viewer. Selection of the Read Only checkbox precludes selection of the Create Memory option.

Optionally, the Create Memory checkbox may be activated.

If selected, the Size of the shared memory segment must be specified. When the sampler process connects to the target system, it will create the shared memory segment if it does not already exist. If it already exists and is of insufficient size, the connection will fail with a diagnostic.

If the Create Memory option is selected, you may also select the Bind option and specify a physical memory address to which to bind the created shared memory segment. If this option is specified, the address will be passed to the **shmbind(2)** system service during connection.

Warning:

Extreme care is recommended when choosing physical addresses. Probing inappropriate physical addresses can cause system instability or crashes.

10
Mapped Memory Window

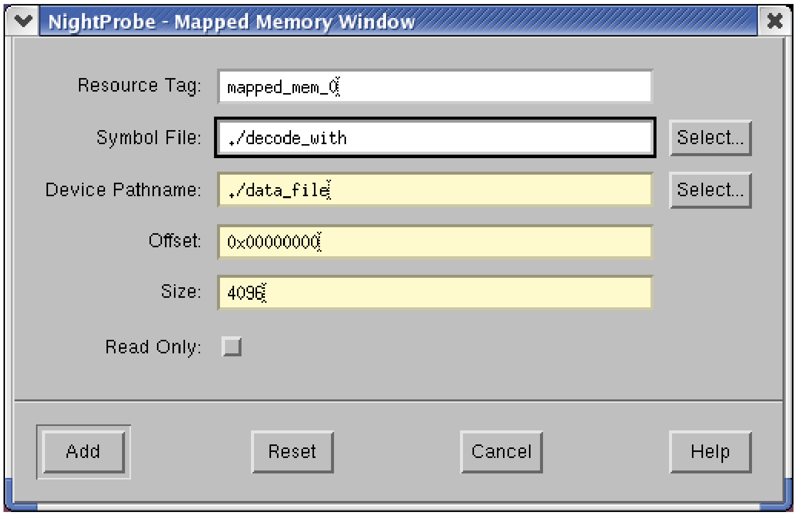
10
Mapped Memory Window

The Mapped Memory window specifies a device or file to monitored or recorded via memory mapping.

It is launched from the Add Mapped Memory... menu option of the Resources icon in the Main window.

This window provides for the selection of five key pieces of information as illustrated in Figure 10-1:

- Resource Tag
- Symbol File
- Device Pathname
- Offset
- Size



Resource Tag: mapped_mem_0

Symbol File: ./decode_with

Device Pathname: ./data_file

Offset: 0x00000000

Size: 4096

Read Only: □

Figure 10-1. The Mapped Memory WIndow

Resource Tag

A Resource Tag is just a short-hand identifier for the resource. It is important as it is used to maintain relationships between variables and resources in the session configuration file so that configurations may be saved and reloaded in the future.

By default, a unique resource tag is assigned when the window is launched.

If the tag name is modified, it will only be adjusted if it conflicts with another resource in which case NightProbe will append an underscore and number to the value of the Resource Tag field when the Add button is pressed and pop up a dialog indicating the conflict and tag adjustment.

Symbol File

A Symbol File is an executable file which contains symbolic and debug information that allows NightProbe to identify and list types that can be associated with artificial variables you create and associate with the mapped memory segment.

A Symbol File is not required for probing a mapped memory segment, but without it, you can only define artificial variables using basic numeric and character types in the Item Definition dialog.

If specified, the Symbol File must exist, be accessible from the host system, and be a valid executable program file.

Device Pathname

A Device Pathname must be specified. It identifies the device or file that will be mapped into the sampler process's memory via the **`mmap(2)`** system service.

The specified pathname does not have to be accessible from the host system.

The Select... button to the right of the Device Pathname text field provides a standard file selection dialog to locate specify the pathname.

Note:

> The pathnames provided in the file selection dialog are relative to the host system, even though the selected file must be accessible from the target system when NightProbe connects to the target system.

System memory may be probed by specifying **`/dev/mem`** as the Device Pathname.

Warning:

Extreme care is recommended in probing system memory.

Offset

An Offset must be specified. By default, the Offset value field is set to zero. This value defines the base offset from the beginning of the mapped memory segment for all variables to be associated with this mapped memory segment. The Offset in the Item Definition dialog is added to the base offset defined here.

Size

The Size of the memory segment must be specified. When NightProbe connects to the target system, if the specified size exceeds the size of the device specified in the Device Pathname text field, the connection will fail with a diagnostic.

Options

The Read Only checkbox may be activated. If activated, the memory mapping will be done in a read-only mode preventing modification of the probed file or device from the Spreadsheet Viewer window.

11
Item Browser Window

11
Item Browser Window

The Item Browser window allows you to navigate the list of variables in a program symbol file and select variables to be probed.

It is launched from the Add Items from Program... menu option on the Probe Items icon in the Main window.

This window has four main control areas as illustrated in Figure 11-1:

- Interactive Variable Tree Browser
- Scope Filter
- Item Filter
- Item To Add

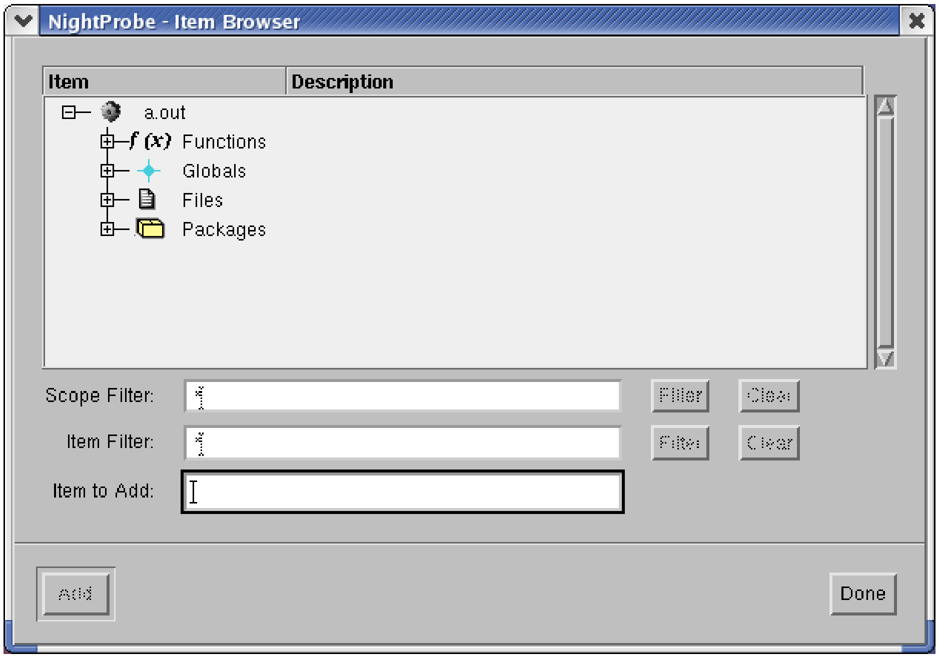


Figure 11-1. The Item Browser Window

Interactive Variable Browser

The interactive browser provides expandable and collapsible lists of entities organized in a tree.

The root of the tree is a list of all programs that have an associated symbol file.

The tree contains scopes, variables, and components of composite variables.

For each node in the tree that has children, the list can be expanded or collapsed by clicking the control box to the left of the icon.

Keyboard traversal is also supported within the tree.

Up Arrow

Pressing the Up Arrow key will cause the node immediately above the current node to be selected unless the top node in a list has been reached. If the top node has been reached, the key has no effect.

Down Arrow

Pressing the Down Arrow key will cause the node immediately below the current node to be selected unless the bottom node in a list has been reached. If the bottom node has been reached, the key has no effect.

Left Arrow

Pressing the Left Arrow key will cause the parent of the current list to be selected. When a root node is reached, the key has no effect.

Right Arrow

Pressing the Right Arrow key will cause automatically expand the current node and cause the first child node to become selected. If the current node is a leaf, the key has no effect.

Spacebar

Pressing the Spacebar toggles the expansion setting of the current node. If the current node has no children, the key has no effect.

For each resource node, there are four main nodes which provide lists of Functions, Globals, Files, and Packages which contain scopes and variables.

Functions

The Functions list is populated with the names of identifiable functions within the symbol file. Functions that contain scopes or variables that can be recorded will have an expandable control box to their left. Fortran subroutines containing common blocks are an example of a function which contains a nested scope.

Globals

The Globals list is populated with variables that are defined in the global scope. Typically these are C and C++ variables declared as `extern`.

Files

The Files list is populated with all identifiable source files within the symbol file. Files may contain scopes, such as packages or functions, and static variables declared outside of functions.

Packages

The Packages list is populated with all identifiable library level Ada packages within the symbol file. Packages contain other packages or variables that can be probed.

Variables

Variables appear by name with boxes as their icon. If a variable is a composite type (an array, structure, or record), it will appear with an expandable control box. All record and structure components are shown when a record or structure is expanded. Array components can also be expanded, but are limited by the resource `Nprobe.text.maxArrayExpansion`, which defaults to 1000.

To add a variable to the list of Probe Items of the Main window, select the variable and click Add. Alternatively, pressing the <enter> key or double-clicking the variable adds the selected variable as well. Pressing the <enter> key also exits the window immediately after adding the item.

Composite variables may be added as a whole, or individual components may be added.

Variables that have been added to the Probe Items list are indicated with a orange icon.

Variables in source files that were compiled without the symbolic debug option (**-g**) or are not elligible to be recorded will not appear in the Item Browser.

Scope Filter

The Scope Filter allows a regular expression to be applied to the entire tree of lists that contain variables. The expression should adhere to the syntax as defined by **regexec(3)**. The expression is not automatically bound to the start or end of a item name. Thus to filter scopes that start with “sched_” you would enter the regular expression “^sched_.*” (without the quotation marks).

Item Filter

The Item Filter allows a regular expression to be applied to all variables in the tree. The expression should adhere to the syntax as defined by **`regexec(3)`**. The expression is not automatically bound to the start or end of a name. Thus to filter variables that end with "_counter" you would enter the regular expression ".*_counter$" (without the quotation marks).

Item filtering stops at the variable level, it is not applied to components of composite variables.

Item to Add

Variables may be directly added to the list of Probe Items of the Main window by typing their fully-expanded name in the Item To Add text field.

See "Variable Name Notation" on page A-1 for information on how to specify a variable in fully-expanded name format.

For array variables, a subscript or slice may be added to the name. See "Array Slices" on page A-2 for information on array slice syntax.

For record or structure variables, a component may be specified.

When the Add button is pressed, the variable is added to the Probe Items list in the Session Overview area of the Main window. If the variable does not match any existing eligible variable in the symbol file, it will be added in a disabled state and its description will indicate "no matching symbol". Variables in the Probe Items list which are in a disabled state have a white icon.

12
Item Definition Window

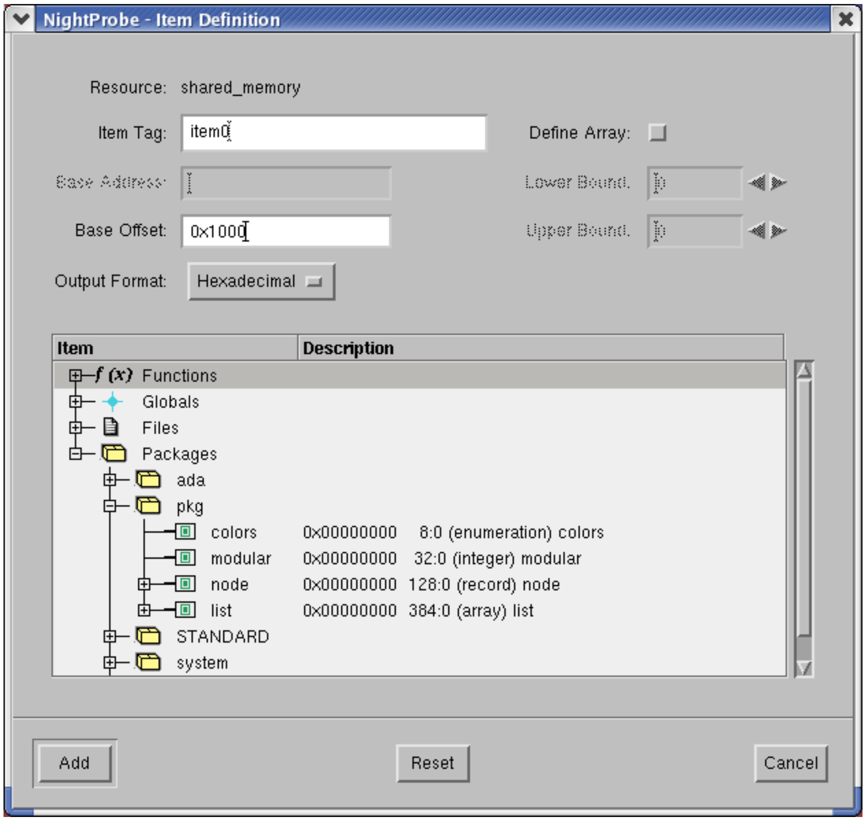
12
Item Definition Window

The Item Definition window allows you to create an artificial variable which is a view into the associated resource.

It is launched from the New Item... menu option on the Probe Items icon in the Main window.

This window has four main configuration areas as illustrated in Figure 12-1:

- Item Tag
- Base Address or Offset
- Output Format
- Interactive Type Tree Browser



Resource: shared_memory

Item Tag: item0

Define Array: □

Base Address:

Lower Bound: 0

Base Offset: 0x1000

Upper Bound: 0

Output Format: Hexadecimal

Item	Description
⊞ f (x) Functions	
⊞ Globals	
⊞ Files	
⊟ Packages	
⊞ ada	
⊟ pkg	
colors	0x00000000 8:0 (enumeration) colors
modular	0x00000000 32:0 (integer) modular
⊞ node	0x00000000 128:0 (record) node
⊞ list	0x00000000 384:0 (array) list
⊞ STANDARD	
⊞ system	

Figure 12-1. The Item Definition Window

Item Tag

An Item Tag is simply a short-hand identifier for the artificial variable being defined by this dialog.

By default, a unique Item Tag is assigned when the window is launched.

If you modify the Item Tag value and it conflicts with an existing Item Tag name, a dialog will pop up and request a change.

Base Address or Offset

The Base Address or Base Offset must be defined for thew artificial variable being defined by this dialog.

For artificial variables associated with program resources, the Base Address should be set to the virtual address you wish to probe within the program.

For artificial variables associated with other resources, the Base Offset should be set to the offset you wish to view within the memory region of the resource. This value added to the Base Offset that was supplied in the associated resource selection window to calculate the final offset of the item in the resource.

Output Format

The Output Format can be selected via the drop-down list. The default setting is Default, which will cause variable values to be displayed in their natural format as described in the following table:

Type Class	Format
Signed Integers	Decimal
Unsigned Integers	Hexadecimal
Real	Exponential
Fixed Point	Exponential
Enumerations	Enumeration Image
Character Arrays	String
Pointers	Hexadecimal

Alternatively, you can select a display format from the list.

Interactive Type Browser

The interactive type browser provides expandable and collapsible lists of entities organized in a tree..

The tree contains scopes, types, and type components.

For each node in the tree that has children, the list can be expanded or collapsed by clicking the control box to the left of the icon.

Keyboard traversal is also supported within the tree.

Up Arrow

> Pressing the Up Arrow key will cause the node immediately above the current node to be selected unless the top node in a list has been reached. If the top node has been reached, the key has no effect.

Down Arrow

> Pressing the Down Arrow key will cause the node immediately below the current node to be selected unless the bottom node in a list has been reached. If the bottom node has been reached, the key has no effect.

Left Arrow

> Pressing the Left Arrow key will cause the parent of the current list to be selected. When a root node is reached, the key has no effect.

Right Arrow

> Pressing the Right Arrow key will cause automatically expand the current node and cause the first child node to become selected. If the current node is a leaf, the key has no effect.

Spacebar

> Pressing the Spacebar toggles the expansion setting of the current node. If the current node has no children, the key has no effect.

For each resource node, there are four main nodes which provide lists of Functions, Globals, Files, and Packages which contain scopes and types.

Functions

> The Functions list is populated with the names identifiable functions within the symbol file. Functions that contain scopes or types that can be recorded will have an expandable control box to their left.

Globals

> The Globals list is populated with basic numeric and character types.

Files

The Files list is populated with all identifiable source files within the symbol file. Files may contain scopes, such as packages or functions, and types declared outside of functions.

Packages

The Packages list is populated with all identifiable library level Ada packages within the symbol file. Packages contain other packages or types.

Types

Types appear by name with boxes as their icon. If a type is a composite type (an array, structure, or record), it will appear with an expandable control box. All record and structure components are shown when a record or structure is expanded. Array components can also be expanded, but are limited by the resource `Nprobe.text.maxArrayExpansion`, which defaults to 1000.

For resources that do not have symbol files associated with them, only the Globals list will be populated with types.

For resources that have symbol files, types will appear in the tree only if the associated source files were compiled with the symbolic debug option (-g). Further, depending on the compiler version, types that were not applied to a variable in such source files may be omitted from the tree.

Each type node contains the type name and a description of the type, including any offset from its composite parent, its bit size, bit offset, and type class (e.g. array, integer, record).

To select a type for the new variable being defined in this dialog, select the desired type node.

A composite type may be added as a whole, or individual components may be added.

If a component of a composite type is added, the offset of the component within the base composite type will be added to the Base Offset value when the new variable is added to the list of Probe Items. This association is maintained in the session configuration..

Thus if a saved configuration is reloaded, and the component's offset had changed due to a modified symbol file, the component offset is adjusted accordingly.

Options

Optionally, you can activate the Define Array checkbox. If selected, the artificial item will be defined as an array of the type that you select in the Interactive Type Tree Browser.

The bounds of the array can be set with the Lower Bound and Upper Bound text fields or via the increase and decrease arrows.

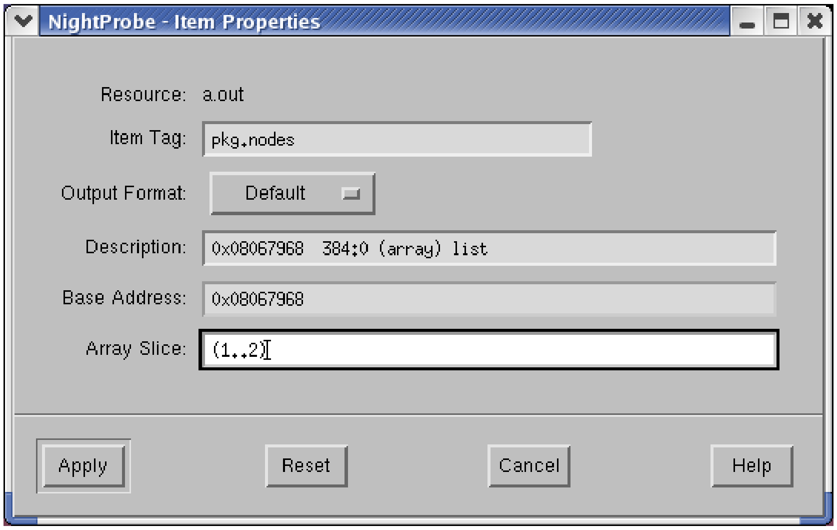
13
Item Properties Window

13
Item Properties Window

The Item Properties window allows you view and modify some attributes of variables listed in the Probe Items of the Main window.

This window has five descriptive areas as illustrated in Figure 13-1:

- Item Tag
- Output Format
- Description
- Base Address or Offset
- Array Slice



Resource: a.out

Item Tag: pkg.nodes

Output Format: Default

Description: 0x08067968 384:0 (array) list

Base Address: 0x08067968

Array Slice: (1..2)

Figure 13-1. The Item Properties Window

Item Tag

The Item Tag, a short-hand identifier for the item, is displayed here.

Output Format

The setting for the default Output Format is displayed in the drop-down list.

This field can be modified to affect how values of this item are displayed.

The Default setting will cause item value to be displayed in its natural format as described in the following table:

Type Class	Format
Signed Integers	Decimal
Unsigned Integers	Hexadecimal
Real	Exponential
Fixed Point	Exponential
Enumerations	Enumeration Image
Character Arrays	String
Pointers	Hexadecimal

Description

The Description of the item includes:

Item Address or Offset

The first field in the Description is the virtual address of a real program variable or the offset of an artificial variable.

For artificial variables which were defined as components of a composite type, the value displayed here may differ from the value displayed in the Base Offset field. For such artificial variables, the value displayed here is the sum of the Base Offset and any offset associated with this component within the outermost enclosing composite type.

Bit Size and Bit Offset

The second field in the Description is the Bit Size and Bit Offset, displayed together, separated by a colon.

Type Class

The third field in the Description is the Type Class, displayed in parentheses. Type classes include:

- Enumeration
- Floating Point
- Fixed Point
- Integer
- Record
- Array
- Character
- Pointer
- (Fortran) Complex

Type Name

The last field in the Description is the Type Name, or short-hand type declaration if no specific type name is available.

Base Address or Offset

The Base Address of program items or the Base Offset of artificial items is shown here.

This value may be modified only for artificial variables created with the Item Definition window.

The Base Offset may differ from the offset displayed in the Description field. This occurs only for artificial items which were defined as components of a composite type. The final offset of the item is defined by the Base Offset plus the component's offset within the outermost containing composite type.

Array Slice

If the item is an array, the current slice information is shown and may be modified.

If a slice is specified, only those array components are sampled.

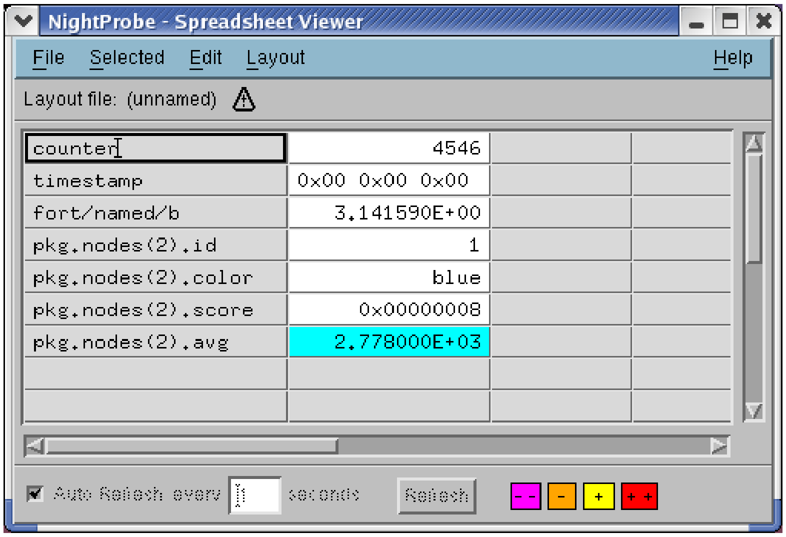
See "Array Slices" on page A-2 for more information on array slice syntax.

14
Spreadsheet Viewer

14
Spreadsheet Viewer

The Spreadsheet Viewer window provides for monitoring variables as well as modifying their values.

.



Layout file: (unnamed)

counter	4546		
timestamp	0x00 0x00 0x00		
fort/named/b	3.141590E+00		
pkg.nodes(2).id	1		
pkg.nodes(2).color	blue		
pkg.nodes(2).score	0x00000008		
pkg.nodes(2).avg	2.778000E+03		

☑ Auto Refresh every [ ] seconds

Figure 14-1. The Spreadsheet Viewer Window

The Spreadsheet Viewer window, shown in Figure 14-1, provides for both viewing and modification of sampled data. It does the following:

- Allows for flexible placement of data values and labels within a spreadsheet with user-defined number and sizes of rows and columns.
- Allows selection and modification of more than one cell at a time.
- Allows for the spreadsheet layout to be saved and restored.
- Displays sampled data after every sample, after a set number of samples, or upon demand.
- Allows modification of data simply by entering the new value into the spreadsheet cell.

For a tutorial on using the spreadsheet viewer, see “Using the Spreadsheet” on page C-4 for an example using an C and Ada program.

The Spreadsheet Viewer window contains

- The Menu Bar (see “Menu Bar” on page 14-2)

- The Layout Configuration Status Area (see "Layout Configuration Status Area" on page 14-10)
- The Spreadsheet Viewing Area (see "Spreadsheet Viewing Area" on page 14-10)
- The Control Area (see "Control Area" on page 14-10)

Menu Bar

The Spreadsheet Viewer window menu bar contains the following menus.

- File
- Selected
- Edit
- Layout
- Help

Each menu is described in the sections that follow.

File

Mnemonic: Alt-F

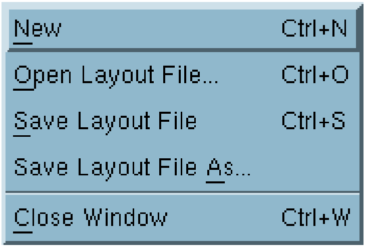


Figure 14-2. File menu

The File menu allows you to load a previously-saved layout configuration, save the current layout configuration to a file, or get a new, clean layout configuration. The File menu also contains the means to close the window. The following paragraphs describe the options on the File menu in more detail.

New
Mnemonic: N
Accelerator: <Control><N>

This option allows you to clear the cells in the spreadsheet and the layout configuration. If you are monitoring a running program, you will not be able to recall the erased information in this window.

Open Layout File...

Mnemonic: O
Accelerator: <Control><O>

This option allows you to open a layout file that was created using the Save Layout File or Save Layout File As options. The layout file saves all information about how the cells in the spreadsheet are used to display the sampled data. You will be presented with a file selection dialog with which to choose a file name.

Save Layout File

Mnemonic: S
Accelerator: <Control><S>

This option allows you to save the spreadsheet layout configuration to the current layout file. If the spreadsheet viewer is not currently associated with a layout configuration file name, this option is the same as Save Layout File As.

Save Layout File As...

Mnemonic: A

This option allows you to save the spreadsheet layout configuration to a file. You will be presented with a file selection dialog with which to choose a file name.

You may also save the image of the currently selected cells as text information to a file by selecting the Save As Text... item from the Selected menu (see “Save as Text” on page 14-7).

Close Window

Mnemonic: C
Accelerator: <Control><W>

Using this option closes this window and removes it from the Output list.

Selected

Mnemonic: Alt-S

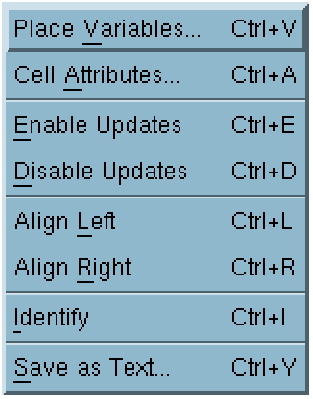


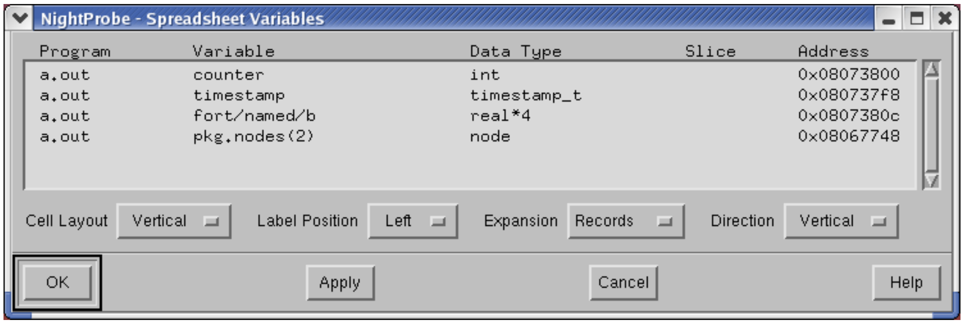
Figure 14-3. Selected menu

The Selected menu operates on a group of spreadsheet cells that have already been selected. Select cells by clicking mouse button 1 with the mouse pointer over the cell, or by dragging the mouse pointer across a rectangle of cells while mouse button 1 is depressed. Selected cells will be highlighted.

Place Variables

Mnemonic: V
Accelerator: <Control><V>

Selecting the Place Variables menu option displays the Spreadsheet Variables window. The figure below shows the Spreadsheet Variables window. This window contains controls to place variable cells onto a spreadsheet.


NightProbe - Spreadsheet Variables

Program	Variable	Data Type	Slice	Address
a.out	counter	int		0x08073800
a.out	timestamp	timestamp_t		0x080737f8
a.out	fort/named/b	real*4		0x0807380c
a.out	pkg.nodes(2)	node		0x08067748

Cell Layout Vertical
Label Position Left
Expansion Records
Direction Vertical

Figure 14-4. The Spreadsheet Variables Window

To use this window, first select a cell in the spreadsheet by clicking on it with the mouse. This will be the starting cell for placing variables.

Next, select the variable or variables you wish to place from the list in the Variable Placement window. You may place more than one variable at a time.

Below the Variable List are three option menus for controlling placement.

The Cell Layout menu is used whenever you place more than one variable at once. It specifies whether to place the variables going down from the starting cell (Vertical layout) or going across the spreadsheet (Horizontal layout).

The Label Position menu controls an optional label cell which will be placed along with the variable cell. The label cell will contain the name of the variable. You can choose None for no label, or a position relative to the Variable cell (Top, Bottom, Left, or Right).

The Expansion menu specifies what to do with variables that represent composite types. Selecting None will place all array elements in a single cell. Selecting Records will automatically expand the record allocating a cell for each individual component. Selecting Arrays will automatically expand array components as individual cells. Selecting Both will expand arrays and records. Expansion is limited to a single level; components which are themselves composites will not be expanded.

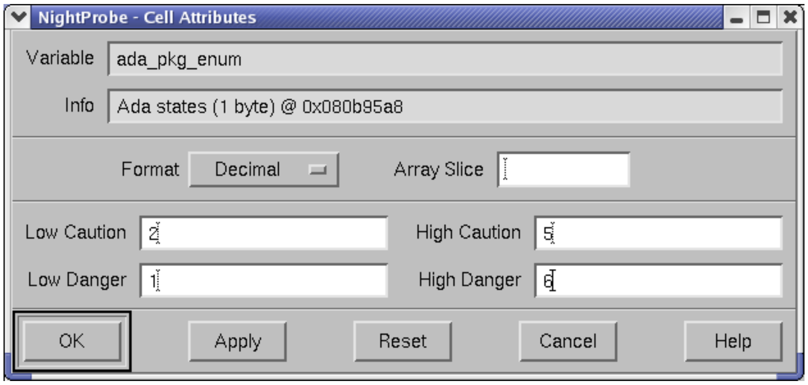
The Direction menu is sensitized when composite expansion is selected. Selecting Horizontal or Vertical will place each composite in its own cell, laid out in the specified direction.

When you have selected your variables and options, click the OK button to place them on the spreadsheet and close the window, or the Apply button to place the variables and leave the window open. The Close button closes the window without placing any variables.

Cell Attributes

Mnemonic: A
Accelerator: <Control><A>

Selecting the Cell Attributes menu option displays the Cell Attributes window. The figure below shows the Cell Attributes window. This window allows you to view and change various attributes associated with a spreadsheet cell. The window is only active when a Variable cell is selected.



NightProbe - Cell Attributes

Variable: ada_pkg_enum

Info: Ada states (1 byte) @ 0x080b95a8

Format: Decimal
Array Slice:

Low Caution: 2
High Caution: 5

Low Danger: 1
High Danger: 6

Figure 14-5. The Cell Attributes Window

Variable and Info

These fields show the variable name and information about the selected cell's variable. They are read-only text fields. To change a cell's variable or to create new variable cells, use the Place Variables option under the Selected menu.

Format

This option menu allows you to choose the output format for the cell.

Array Slice

This field allows you to specify the array indices, if the variable is an array, to display in the cell. You may specify a single index number or a range of numbers such as `3..7` or `3:7` for elements 3 through 7, inclusive. For more information about array slices, see "Array Slices" on page A-2.

Low Caution, High Caution, Low Danger, High Danger

These fields specify limits for the specified variable. They are only appropriate for use with scalar types. When the value in the cell goes outside these boundaries, the cell's background color will change. You can define the colors of these cells with resources described in "NightStar Resources" on page D-2.

Clicking the OK button applies any changes you have made to the cell and closes the window. Clicking the Apply button applies the changes without closing the window. Clicking the Cancel button closes the window without making any changes (this button will be labeled Close if no changes were made).

Enable Updates

Mnemonic: E
Accelerator: <Control><E>

Updates the selected cells when new samples are displayed. This reverses the action of the Disable Updates selection.

Disable Updates

Mnemonic: D
Accelerator: <Control><D>

Does not update the selected cells when new samples are displayed. These cells have a darker background color than enabled cells. You would use this option to hold on to a data value in the display while allowing the sampler to continue running and updating other values.

Align Left

Mnemonic: L
Accelerator: <Control><L>

Data values in the selected cells will be aligned with the left edge of the cell.

Align Right

Mnemonic: R
Accelerator: <Control><R>

Data values in the selected cells will be aligned with the right edge of the cell.

Identify

Mnemonic: I
Accelerator: <Control><I>

Displays the variable names or addresses with which the selected cells are associated. The next update will revert to displaying the data values.

Save as Text

Mnemonic: S
Accelerator: <Control><Y>

Writes as text information to a file the image of the currently selected cells. You will be presented with a file selection dialog with which to choose a file name.

You may also save the spreadsheet layout configuration to a file by selecting the Save Layout File As... item from the File menu (see “File” on page 14-2).

Edit

Mnemonic: Alt-E

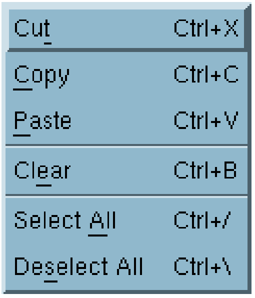


Figure 14-6. Edit menu

The Edit menu provides the means to perform some editing operations on the cells and the layout configuration.

Cut
Mnemonic: T
Accelerator: <Control><X>

Removes the layout configuration information from the selected cells and stores that information in the layout clipboard. The selected cells are cleared.

Copy
Mnemonic: C
Accelerator: <Control><C>

Copies the layout configuration information from the selected cells and stores that information in the layout clipboard. The selected cells are unaffected.

Paste
Mnemonic: P
Accelerator: <Control><V>

Inserts the contents of the layout clipboard at the current selection point. The layout clipboard retains its information and can be used again.

Clear
Mnemonic: E
Accelerator: <Control><B>

Clears the selected cells.

Select All
Mnemonic: A
Accelerator: <Control></>

Puts all cells into the "selected" state for other operations.

Deselect All
Mnemonic: S
Accelerator: <Control><\>

Puts all cells into the "unselected" state.

Layout

Mnemonic: Alt-L

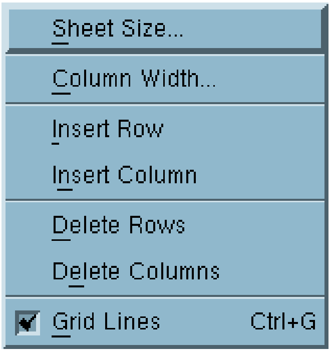


Figure 14-7. Layout menu

The Layout menu provides controls for organizing the display area into a rectangular grid of spreadsheet cells.

Sheet Size
Mnemonic: S

Displays a dialog that allows you to specify the number of rows and columns in the spreadsheet.

Column Width
Mnemonic: C

Displays a dialog that allows definition of the width (in character positions) of the currently selected columns.

Insert Row
Mnemonic: I

Inserts one row above the current selection point.

Insert Column
Mnemonic: N

Inserts one column to the left of the current selection point.

Delete Rows
Mnemonic: D

Deletes the selected rows.

Delete Columns
Mnemonic: E

Deletes the selected columns.

Grid Lines
Mnemonic: G

Enables or disables the lines delineating the spreadsheet cells by clicking on the toggle button.

Help

Mnemonic: Alt-H

The Help menu operates exactly like the menu provided in the Main window. It lists a number of topics on which help is available, and selecting any topic will display a help window. See "Getting Help" on page 2-2 for details.

Layout Configuration Status Area

The layout configuration status area displays the file name of the layout file, if any has been specified. It also indicates via an icon at the end of the name if there are unsaved modifications to the layout configuration.

Spreadsheet Viewing Area

The spreadsheet viewing area is composed of rows and columns of spreadsheet cells. Each cell can contain either a text label or the contents of a monitored data location. Enter text labels merely by selecting the cell and typing the label. Use the Place Variables menu option (from the Selected menu) to associate a selected cell with a data location.

Data values in the spreadsheet are updated according to the specifications of the Control Area (see "Control Area" on page 14-10).

You may use the scroll bars below and to the right of the viewing area to see cells that are not in the current display window.

Once NightProbe has been connected to the target system, you can use the spreadsheet to modify data values. To modify a variable's value, click on the variable cell, type a new value, and press the <Enter> key. Values may be entered as decimal numbers, octal numbers when preceded by `0`, hexadecimal numbers when preceded by `0x`, enumeration constants as identifiers, or character strings when enclosed in double quotes (").

Control Area

The control area appears at the bottom of the window. It allows you to control when new information is added to the viewing area. In addition, it contains a legend indicating the two caution and two danger colors.

Auto Refresh

The Auto Refresh checkbox and text entry field allow you to display every *n*th sample, where *n* is a value you select. This is useful if you want to monitor a program while it is running but the sampler is recording values so fast they cannot be seen. (See also “Invoking NightProbe” on page 2-1 and “NightStar Resources” on page D-2.)

Refresh

The Refresh button gets the most recent sample taken and displays it in the Spreadsheet Viewer window. The Refresh button does not cause the sampler to take a new sample or record a sample to a file.

Cell Color Legend

These four squares are a legend indicating the colors used when the value in a cell exceeds its defined limits.

– –	represents Low Danger
–	represents Low Caution
+	represents High Caution
+ +	represents High Danger

Limits for a cell can be set using the Cell Attributes window. You can define the colors of these cells with resources described in “NightStar Resources” on page D-2.

15
NightProbe API

15
NightProbe API

The NightProbe Application Programming Interface provides a basic interface to the data produced by NightProbe.

This API can be used with data recording output generated by NightProbe using the To File and To Program output methods (see “File Specification” on page 6-1 and “To Program” on page 6-9).

The following sections describe the data structures and functions that comprise the NightProbe API.

Sample programs using these data structures and functions are also provided (see “Sample Programs” on page 15-12).

NightProbe Data Format

This section describes the general format of data generated by NightProbe sampling. This format is used when you select either the To File or To Program output method (see “File Specification” on page 6-1 and “To Program” on page 6-9).

The NightProbe Application Programming Interface (see “NightProbe Application Programming Interface” on page 15-2) allows you to open a previously recorded file and decode the individual data items or to consume the data as it is being generated by NightProbe. In either case, the incoming data is referred to as a datastream. In the case of To File, you open the file and use the resultant file descriptor with the following API calls to decode the data. In the case of To Program, a user program is launched from Night-

Probe and its **stdin** file descriptor is set to the read end of a socket or pipe. The user program then specifies **stdin** to the following API calls to decode the data.

The following diagram describes the general layout of a datastream:

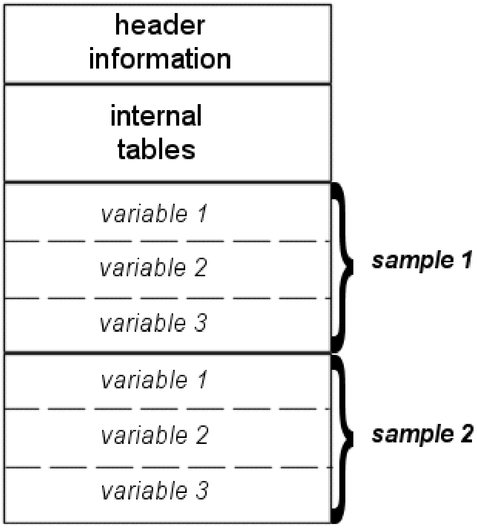


Figure 15-1. Structure of NightProbe datastream

The API calls below provide a simple interface for obtaining information about the programs from which the data was obtained, information about the variables within those programs, and individual data *sample*s.

NightProbe Application Programming Interface

The NightProbe Application Programming Interface consists of a number of data structures (see “Data Structures” on page 15-2) and functions (see “Functions” on page 15-6).

Data Structures

The following data structures are part of the NightProbe Application Programming Interface:

- `np_handle` (see “np_handle” on page 15-3)
- `np_header` (see “np_header” on page 15-3)
- `np_process` (see “np_process” on page 15-3)
- `np_item` (see “np_item” on page 15-4)
- `np_type` (see “np_type” on page 15-5)

See “Functions” on page 15-6 for information about the functions available in the NightProbe API.

np_handle

np_handle is a unique integer value denoting a single NightProbe datastream.

```
typedef int np_handle;
```

See “Data Structures” on page 15-2 for other data structures included in the NightProbe API.

np_header

np_header is a structure which is used to describe the processes and items from which data in the NightProbe datastream originates. This information is needed to interpret the sample data returned by np_read().

```
typedef struct {
   int           num_items;
   int           num_processes;
   int           sample_size;
   np_process  * processes;
   np_item     * items;
} np_header ;
```

See “Data Structures” on page 15-2 for other data structures included in the NightProbe API.

SEE ALSO

- “np_read()” on page 15-8

np_process

np_process is a structure which contains information about a particular process from which real-time data originates.

```
typedef struct np_process np_process;
struct np_process {
   int          pid;
   char       * name;
   np_process * link;
};
```

See “Data Structures” on page 15-2 for other data structures included in the NightProbe API.

np_item

np_item is a structure that describes a single data item present in the NightProbe datastream.

```
typedef struct np_item np_item;
struct np_item {
   char        * name;     // name of item
   int           offset;   // byte offset within each sample
   unsigned      size;     // atomic size in bytes
   unsigned      count;    // number of atoms
   np_type       type;     // data type
   unsigned      event_id; // NightTrace event ID for item
   np_process  * process;  // process info
   np_item     * link;     // next item pointer
};
```

The item occupies count instances of size bytes beginning at offset within the sample data returned by np_read().

See "Data Structures" on page 15-2 for other data structures included in the NightProbe API.

SEE ALSO

- "np_read()" on page 15-8
- "np_type" on page 15-5

np_type

The `np_type` enumeration in the `np_item` structure may be used (along with `size`) in order to determine an appropriate format for displaying a value from the sample buffer.

```
typedef enum np_type_code {
   NP_VOID_TYPE,                  /* void                          */
   NP_CHAR_TYPE,                  /* signed byte character         */
   NP_UNSIGNED_CHAR_TYPE,         /* unsigned byte character       */
   NP_SHORT_INT_TYPE,             /* signed short int              */
   NP_UNSIGNED_SHORT_INT_TYPE,    /* unsigned short int            */
   NP_INT_TYPE,                   /* signed int                    */
   NP_UNSIGNED_INT_TYPE,          /* unsigned int                  */
   NP_LONG_INT_TYPE,              /* signed long int               */
   NP_UNSIGNED_LONG_INT_TYPE,     /* unsigned long int             */
   NP_FLOAT_TYPE,                 /* single precision float        */
   NP_DOUBLE_TYPE,                /* double precision float        */
   NP_LONG_DOUBLE_TYPE,           /* long double precision float   */
   NP_SHORT_LOGICAL_TYPE,         /* short logical (boolean)       */
   NP_LOGICAL_TYPE,               /* logical (boolean)             */
   NP_COMPLEX_TYPE,               /* Fortran complex type          */
   NP_DOUBLE_COMPLEX_TYPE,        /* Fortran double complex        */
   NP_POINTER_TYPE,               /* Pointer to unspecified type   */
   NP_FIXED_POINT_TYPE,           /* fixed point                   */
   NP_EXCEPTION_TYPE,             /* exception                     */
   NP_STRUCTURE_BYTES             /* structure bytes               */
}  np_type ;
```

See “Data Structures” on page 15-2 for other data structures included in the NightProbe API.

SEE ALSO

- “np_item” on page 15-4

Functions

The following functions are part of the NightProbe Application Programming Interface:

- `np_open()` (see "np_process" on page 15-3)
- `np_avail()` (see "np_avail()" on page 15-7)
- `np_read()` (see "np_read()" on page 15-8)
- `np_close()` (see "np_close()" on page 15-9)
- `np_close()` (see "np_format()" on page 15-10)
- `np_error()` (see "np_error()" on page 15-11)

np_open()

`np_open()` is used to open and initialize an input NightProbe datastream on an open file descriptor.

SYNTAX

```
int np_open (int fd, np_header *header, np_handle *handle);
```

PARAMETERS

fd file descriptor associated with the file created using the File Specification output method (see "File Specification" on page 6-1) which contains the data recording output

If data recording output is streamed directly from NightProbe using the Program output method (see "To Program" on page 6-9), *fd* should be set to the **`stdin`** file descriptor, `0`.

header structure to contain information describing the processes from which the NightProbe data originates, as well as the number, names and types of the items appearing in the NightProbe datastream

handle a unique value denoting the open NightProbe datastream

RETURN VALUES

0 indicates successful completion

-1 indicates a failure

handle contains a value which may be passed to `np_error()` to obtain a diagnostic message describing the failure

See “Functions” on page 15-6 for other functions included in the NightProbe API.

SEE ALSO

- “np_header” on page 15-3
- “np_handle” on page 15-3
- “np_error()” on page 15-11

np_avail()

`np_avail()` is used to check a NightProbe datastream for available data items.

SYNTAX

```
int np_avail (np_handle handle);
```

PARAMETERS

handle value (obtained from `np_open()`) which identifies the NightProbe datastream of interest

RETURN VALUES

0 if data is not currently available on the NightProbe datastream and `np_read()` would block

> 0 if data is currently available for `np_read()`

-1 indicates a failure

If *handle* is non-zero, `np_error()` may be called to obtain a diagnostic message describing the failure.

See “Functions” on page 15-6 for other functions included in the NightProbe API.

SEE ALSO

- “np_open()” on page 15-6
- “np_read()” on page 15-8
- “np_error()” on page 15-11

np_read()

Read a single data sample from the NightProbe datastream.

SYNTAX

```
int np_read (np_handle handle, void *sample);
```

PARAMETERS

handle — value (obtained from `np_open()`) which identifies the NightProbe datastream of interest

sample — upon successful completion, *sample* contains the NightProbe entire sample data

To get at individual data items, use the information from the `np_header` structure returned from `np_open()`. For each item, retrieve the appropriate number of bytes (as specified by `size` in the `np_item` structure associated with that item) offset from the beginning of the sample buffer (as specified by `offset` in the `np_item` structure associated with that item)

See "Sample Programs" on page 15-12 for examples.

RETURN VALUES

> 0 — value represents the number of bytes in the sample obtained

0 — if end-of-file (EOF) was encountered on the NightProbe datastream

-1 — indicates a failure

If *handle* is non-zero, `np_error()` may be called to obtain a diagnostic message describing the failure.

NOTE

`np_read()` will block waiting for data to become available on the datastream if data is not immediately available. If time is critical and a blocking read is not desired, use `np_avail()` to first check if data is available prior to reading.

See "Functions" on page 15-6 for other functions included in the NightProbe API.

SEE ALSO

- "np_header" on page 15-3

- "np_item" on page 15-4
- "np_open()" on page 15-6
- "np_avail()" on page 15-7
- "np_error()" on page 15-11

np_close()

Close a NightProbe datastream.

SYNTAX

```
void np_close (np_handle handle);
```

PARAMETERS

handle value (obtained from `np_open()`) which identifies the NightProbe datastream of interest

Upon completion, *handle* no longer refers to an open NightProbe datastream.

NOTE

No further diagnostic messages are available from `np_error()` after calling `np_close()`.

Furthermore, the file descriptor passed to `np_open()` remains open after the `np_close()` call. The NightProbe datastream is logically closed, but the associated file descriptor remains open. **`close(2)`** must be called to close the file descriptor as well, if desired.

See "Functions" on page 15-6 for other functions included in the NightProbe API.

SEE ALSO

- "np_open()" on page 15-6
- "np_error()" on page 15-11

np_format()

Return an allocated string representation of the specified np_item value from the given sample. The caller is responsible for freeing the memory associated with the returned string once it is no longer needed.

SYNTAX

```
char * np_format (np_handle handle,
                  np_item * i,
                  void * sample,
                  int which);
```

PARAMETERS

handle value (obtained from `np_open()`) which identifies the NightProbe datastream of interest

i a pointer to an `np_item` descriptor denoting a single item within a data sample. The `np_item` is part of the `np_header` obtained from the previous call to `np_open()`.

sample a pointer to the contents of a single sample obtained from a call to `np_read()`

which for items with multiple atoms (i.e. *i->count* > 1), *which* determines the atom to be formatted. A *which* value of zero indicates the first atom for the item.

RETURN VALUES

non-NULL value represents a textual representation of the specified data in a format based on the `np_type` of the item

NULL a parameter was invalid, or the NightProbe API was unable to allocate memory for the result. np_error may be called to obtain a diagnostic message describing the failure.

See "Functions" on page 15-6 for other functions included in the NightProbe API.

SEE ALSO

- "np_header" on page 15-3
- "np_item" on page 15-4
- "np_error()" on page 15-11

np_error()

Return a diagnostic message describing the most recent failure encountered by a prior call to `np_open()`, `np_avail()`, or `np_read()`.

SYNTAX

```
char * np_error (np_handle handle);
```

PARAMETERS

handle	value (obtained from `np_open()`) which identifies the NightProbe datastream of interest

See “Functions” on page 15-6 for other functions included in the NightProbe API.

SEE ALSO

- “np_open()” on page 15-6
- “np_avail()” on page 15-7
- “np_read()” on page 15-8

Sample Programs

The following programs are given as examples of how to use the NightProbe Application Programming Interface (see "NightProbe Application Programming Interface" on page 15-2).

program_output_test.c

This program uses the NightProbe API to process a NightProbe data sample.

See "program_output_test.c" on page 15-12.

program_output_fbs_test.c

This program uses the NightProbe API to process a NightProbe data sample but uses a frequency-based scheduler in order to coordinate data recording activity so as to minimize interference with the probed application.

See "program_output_fbs_test.c" on page 15-15.

program_output_test.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#include <nprobe.h>



int    cycles   = 0 ;
int    overruns = 0 ;
char * sample;


//
// Perform the work of consuming a single Data Recording sample from NightProbe.
//
int
work (FILE * ofile, np_handle h, np_header * hdr) {
   np_item * i;
   int status;
   int n;
   char * ptr;


   // Read one sample, which may contain data for multiple processes
   // and variables.
   status = np_read (h, sample);
   if (status <= 0)  {
      return status;
   }

   cycles++ ;

   fprintf (ofile, "\n");
   for (i=hdr->items; i; i=i->link) {
      fprintf (ofile, "item: %25s :", i->name);
      ptr = sample + i->offset;
      for (n=0; n<i->count; ++n) {

         //
         // Note that this simple example assumes type/format from
         // the size of the data item.  The 'i->type' field should
         // be taken into account for a more accurate means of
         // determining the data format.
         //
         switch (i->size) {
         case 1:
            fprintf (ofile, " 0x%1x", ((char*)ptr)[n]);
            break;
         case 2:
            fprintf (ofile, " 0x%1x", ((unsigned short*)ptr)[n]);
            break;
         case 4:
            fprintf (ofile, " 0x%1x", ((unsigned*)ptr)[n]);
            break;
         case 8:
            fprintf (ofile, " %lf", ((double*)ptr)[n]);
            break;
         default:
            fprintf (ofile, " <size=%d>", i->size);
         }
      }
      fprintf (ofile, "\n");
   }
   fflush (ofile) ;

   return 1 ;
}




int
main (int argc, char *argv[])
{
   np_handle h;
   np_header hdr;
   np_process * p;
   np_item * i;
   int fd;
   int status;
   FILE *   ofile = stdout ;


   fd = 0 ; // stdin

   status = np_open (fd, &hdr, &h);
   if (status) {
      fprintf (stderr, "np_open: \"%s\"\n", np_error(h));
      exit(1);
   }

   sample = (char *) malloc(hdr.sample_size);
   if (sample == NULL) {
      fprintf (stderr, "insufficient memory to allocate sample buffer\n");
      exit(1);
   }

   for (p=hdr.processes; p; p=p->link) {
      fprintf (ofile, "process: %s (%d)\n", p->name, p->pid);
   }
   fprintf (ofile, "\n");

   for (i=hdr.items; i; i=i->link) {
      fprintf (ofile, "item: %s (%s) size=%d count=%d type=%d\n",
              i->name, i->process->name, i->size, i->count, i->type);
   }
   fprintf (ofile, "\n");

   for (;;) {
      status = work (ofile, h, &hdr)  ;
      if (status <= 0) break ;
   }

   fprintf (ofile, "Data Recording done: %d cycles fired, %d overruns\n",
            cycles, overruns) ;

   if (ofile != stdout) {
      fclose (ofile) ;
   }

   if (status < 0) {
      fprintf (stderr, "np_read: \"%s\"\n", np_error(h));
   }

   np_close (h);

   // At this point, file descriptor 0 remains open, but is no
   // longer a NightProbe Data File/Stream.
}
```

program_output_fbs_test.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#include <nprobe.h>



int    cycles   = 0 ;
int    overruns = 0 ;
char * sample;


//
// Perform the work of consuming Data Recording samples from NightProbe.
// This function is called once every time the fbswait() system call returns
// successfully.
//
int
work (FILE * ofile, np_handle h, np_header * hdr) {
   np_item * i;
   int status;
   int n;
   char * ptr;


   //
   // 0, 1, or >1 trigger events may haveoccurred since we last work()ed.
   // Check whether data is available, and process it as long as new
   // data is already available within this work cycle.
   //
   // A more sophisticated program would limit the number of np_read() calls
   // per work cycle based upon how much time is left in the current cycle.
   //
   while (np_avail (h)) {

      // Read one sample, which may contain data for multiple processes
      // and variables.
      status = np_read (h, sample);
      if (status <= 0)  {
         return status;
      }

      cycles++;

      fprintf (ofile, "\n");
      for (i=hdr->items; i; i=i->link) {
         fprintf (ofile, "item: %25s :", i->name);
         ptr = sample + i->offset;
         for (n=0; n<i->count; ++n) {

            //
            // Note that this simple example assumes type/format from
            // the size of the data item.  The 'i->type' field should
            // be taken into account for a more accurate means of
            // determining the data format.
            //
            switch (i->size) {
            case 1:
               fprintf (ofile, " 0x%1x", ((char*)ptr)[n]);
               break;
            case 2:
               fprintf (ofile, " 0x%1x", ((unsigned short*)ptr)[n]);
               break;
            case 4:
               fprintf (ofile, " 0x%1x", ((unsigned*)ptr)[n]);
               break;
            case 8:
               fprintf (ofile, " %lf", ((double*)ptr)[n]);
               break;
            default:
               fprintf (ofile, " <size=%d>", i->size);
            }
         }
         fprintf (ofile, "\n");
      }
      fflush (ofile);
   }

   return 1;
}




int
main (int argc, char *argv[])
{
   np_handle h;
   np_header hdr;
   np_process * p;
   np_item * i;
   int fd;
   int status;
   FILE *   ofile = stdout ;


#ifdef linux
   if (!fbsavail()) {
      fprintf (ofile, "fbsavail() reports No FBS on this target\n") ;
      fclose (ofile) ;
      exit (1) ;
   }
#endif



   fd = 0 ; // stdin

   status = np_open (fd, &hdr, &h);
   if (status) {
      fprintf (stderr, "np_open: \"%s\"\n", np_error(h));
      exit(1);
   }

   sample = (char *) malloc(hdr.sample_size);
   if (sample == NULL) {
      fprintf (stderr, "insufficient memory to allocate sample buffer\n");
      exit(1);
   }

   for (p=hdr.processes; p; p=p->link) {
      fprintf (ofile, "process: %s (%d)\n", p->name, p->pid);
   }
   fprintf (ofile, "\n");

   for (i=hdr.items; i; i=i->link) {
      fprintf (ofile, "item: %s (%s) size=%d count=%d type=%d\n",
              i->name, i->process->name, i->size, i->count, i->type);
   }
   fprintf (ofile, "\n");

   for (;;) {
      //
      // We wait till the Concurrent FBS wakes us up at the time which is
      // appropriate for performing data recording.  This program must be
      // scheduled on the FBS, but doing so allows the scheduling of data
      // recording activity at a time that won't disturb other critical
      // application cycles.
      //
      int stat = fbswait() ;

      // Diagnose the return value from fbswait()
      if (stat < 0) {
         switch (stat) {
         case -1:
            if (errno == ENOENT) {
               fprintf (ofile,
                        "%s has been removed from the scheduler\n", argv[0]) ;
            } else {
               fprintf (ofile, "fbs_wait(3) failed on cycle %d: "
                             "errno is %d (%s)\n",
                        cycles, errno, strerror (errno)) ;
            }
            break ;
         default:
            fprintf (ofile, "fbs_wait(3) returned unexpected %d on cycle %d\n",
                     stat, cycles) ;
            break ;
         }

         break ;
      }

      switch (stat) {
      case 0:
         break ;
      case 1:
         fprintf (ofile, "fbstrig(2) caused sim to fire: cycle %d\n", cycles) ;
         break ;
      case 2:
         fprintf (ofile, "soft overrun %d detected on cycle %d\n",
                  ++overruns, cycles) ;
         break ;
      }

      status = work (ofile, h, &hdr);
      if (status <= 0) {
         break;
      }
   }

   fprintf (ofile, "Data Recording done: %d cycles fired, %d overruns\n",
            cycles, overruns) ;

   if (ofile != stdout) {
      fclose (ofile) ;
   }

   if (status < 0) {
      fprintf (stderr, "np_read: \"%s\"\n", np_error(h));
   }

   np_close (h);

   // At this point, file descriptor 0 remains open, but is no
   // longer a NightProbe Data File/Stream.
}
```

A
Variables

Variable Name Notation

Variable names may be used to identify memory addresses in C and Fortran and Ada (using the MAXAda compiler) programs. NightProbe accepts and displays variables with the following syntax.

Syntax

```
["file".][/common/][scope.] ... name[(array_slice)]
0xaddress[:n]
```

Parameters

file The source file name enclosed in double-quotes.

common The common block name enclosed in slashes (or // for unnamed common blocks)

scope The name of the scope. Includes the names of enclosing functions, packages, or composite variables. Each one is separated from the next by a dot (`.`). (See “Composite Types” on page A-2 for information about composite types.)

name The name of the variable. The variable may be either a scalar, an array, a structure or record, or a component of a variable of a composite type.

array_slice An index representing a single array element or an index range representing an array slice. *Array_slices* must be enclosed in either parentheses `()` or square brackets `[]`. (See “Array Slices” on page A-2 for information about array slices.)

address A memory address beginning with a number. If it begins with `0`, it is treated as an octal address. If it begins with `0x`, it is treated as a hexadecimal address.

n An integer representing the size in bytes. It has a colon prefix.

Note that *name*, *name*(), and *name*[] all refer to the entire array.

For some examples using variables in NightProbe, see “Variable Browsing” on page C-2 and “Creating a View into the Device” on page C-14.

Composite Types

To NightProbe, arrays, C structures and unions and Ada records are *composite types*.. Composite objects may be recorded as a whole or individual components within the object may be recorded.

Array Slices

Array slices identify a single element or a range of elements in an array. You select one array element in a manner just like you would use in your program:

```
var (5)
```

Some programming languages use brackets instead of parentheses, as in

```
var [5]
```

NightProbe accepts either convention.

In some cases, it is appropriate to select a range of elements. These elements must be contiguous, and all must lie within the stated bounds of the original array declaration. You specify a range by providing the first and last items that you wish to select. The following syntaxes are all equivalent and may be used with programs of any language.

array_name (*first_item* : *last_item*)
array_name [*first_item* : *last_item*]
array_name (*first_item* .. *last_item*)
array_name [*first_item* .. *last_item*]

where:

array_name — The name of an array.

first_item — A valid array index, greater than or equal to the lower bound of the array and less than or equal to *last_item*.

last_item — A valid array index, less than or equal to the upper bound of the array and greater than or equal to *first_item*.

For example, in Fortran the array declaration

```
integer*4    var (10)
```

declares an array of ten integers with indices 1 through 10. To specify an array slice containing the first five elements, you would use

```
var (1:5)
```

C programs use 0 as the lower bound of all arrays. The declaration

```
int  var [10];
```

also declares an array of ten integers, but with indices 0 through 9.

The equivalent array slice would be

```
var (0:4)
```

Of course, if you are a C programmer you would probably use brackets:

```
var [0:4]
```

and you might prefer the Ada range notation:

```
var [0..4]
```

These last three examples are all equivalent.

In C and Ada, the rightmost subscript of a multi-dimensional array changes most quickly. In Fortran, the leftmost subscript of a multi-dimensional array changes most quickly. Array slices must identify elements that are contiguous in memory. For example, for an 8 by 8 array:

C

Specify `var[1][2]` to refer to the memory location right after `var[1][1]`. The following array slice is valid: `var[3][1:5]`.

Fortran

Specify `var(2,1)`to refer to the memory location right after `var(1,1)`.The following array slice is valid: `var(1:5,3)`.

Ada

Specify `var(1,2)`to refer to the memory location right after `var(1,1)`. The following array slice is valid: `var(3,1..5)`.

Variable Eligibility for Program Resources

Any process on any processor can be a target program for data recording and monitoring.

As stated before, variable names may be used to identify memory addresses in programs. If you wish to identify memory locations by variable name, the target program file must contain symbol table and debug information. Use the **`-g`** compiler option to generate debug information, and do not use the **`-s`** linker option that strips symbol table information from the executable program file.

Any fixed (static) address in a program can be monitored and recorded. The following text lists eligible variables by language.

C

- Variables typed `static`

- Global variables declared outside all functions

Fortran

- Variables typed `static` or `save`
- Variables initialized in a `data` statement
- Variables placed in a `common` block

Ada

The following criteria are used to determine if an Ada data object is eligible for data monitoring/recording:

- The compilation unit containing the object must be a library-level package specification or body. Objects declared in nested packages inside a library-level package are also eligible.
- The object must not be declared in a generic or in the instantiation of a generic.
- The object must have a size and representation which is statically determined at compile time.
- The object may be declared in a library-level package marked with pragma `SHARED_PACKAGE`.

The following Ada data types are eligible for data monitoring/recording:

- Any integer, fixed-point or floating-point type or subtype.
- Any character, Boolean or enumeration type or subtype.
- Access types.
- Array and record types (for records with variant parts, only components that have a statically determined component offset are eligible).

NOTE

Task types and variables declared in Ada procedures or tasks, or objects in an access type's collection, are allocated dynamically, and are, therefore, ineligible for data monitoring/recording.

B
Keyboard Traversal

Keys, Accelerators, and Mnemonics

NightProbe uses certain key combinations as shortcuts for displaying menus and selecting menu items. These key combinations are called *accelerators* and *mnemonics*. Each window has its own set of accelerators and mnemonics that are active only while the keyboard focus is in that window. However, the keyboard focus does not have to be in any particular field of the window to use accelerators and mnemonics. This manual shows the supplied mnemonics and accelerators associated with a menu or menu item. However, users can alter this behavior with resources. See “NightStar Resources” on page D-2 for details.

- Menus can be displayed with mnemonics.

  Menus can be displayed from the keyboard by typing <Alt>+*mnemonic*. Each of the main windows has a menu bar near the top of the window. The different menus are labeled. For example, the Main window has a Timer menu. If you look at the Timer menu, you can see that the T is underlined. T is the mnemonic for the Timer menu. That means that, in addition to displaying the Timer menu by clicking on it with mouse button 1, you can also display it with <Alt>+t (hold down <Alt> and press t).

  If you decide you don’t want to select any of the menu items, you can make the menu go away by typing <Esc> or by clicking somewhere else.

- Menu items can be selected with mnemonics.

  Once a menu is displayed, you can select a menu item by typing only the mnemonic for that item. The mnemonics for the menu items are underlined, just as the mnemonics for the menus are underlined. To select a menu item by using its mnemonic, just press the key.

- Menu functions can be invoked with accelerators.

  Some commonly used menu items have accelerator keys. The functions associated with these menu items can be invoked directly, without displaying the menu, by pressing the accelerator keys. The accelerator keys for a particular menu item are listed next to the item in the menu.

  The accelerator keys are often a combination of a control key plus a letter, such as Ctrl+O. To type Ctrl+O, hold down the control key and press o.

In addition to mnemonics and accelerators, there are also special keys used for navigation within and among windows and fields. These keys include Tab, Shift Tab, Home, End, Page Up, Page Down and the arrow keys. The documentation of these keys is beyond the scope of this chapter. For more information about keys, see the *OSF/Motif User's Guide*.

There are many special keys used to edit text input areas.

Table 18-1 contains a list of some of NightProbe's accelerators and the resulting actions; where applicable, it indicates the menu items for which the accelerators provide shortcuts. Note that you can define additional accelerators through the use of X resources (refer to the **X(1)** system manual page).

Table 18-1. NightProbe Accelerators

Accelerator	Menu Item	Action
<Control> <S>	File Ì Save Config File	Saves the configuration data in the file that is associated with the current window
<Control> <Q>	File Ì Exit	Exits **nprobe**
<Control> <A>	Resource Ì Add Item	Opens the Item Browser window.
<Control> <I>	Resource Ì New Item	Opens the Item Definition window.
<Control> <T>	Control Ì Connect	Connects to target system and resources.
<Control> <D>	Control Ì Disconnect	Disconnects from target system and resources.
<Control> <R>	Control Ì Start	Starts iterative sampling.
<Control> <P>	Control Ì Stop	Stops iterative sampling.
<Control> <L>	Control Ì Sample	Obtains a single sample
<F1>		Displays help for the component that currently has the focus
<Shift> <F1>		Performs same function as Help -> On Context

C
Tutorials

This section contains two separate tutorials which demonstrate the commonly used features of NightProbe:

- "Probing Programs Tutorial" on page C-1, which probes a program written in C and Ada.
- "Probing Devices Tutorial" on page C-12, which probes a PCI device

Probing Programs Tutorial

This tutorial demonstrates some of the commonly used features of NightProbe including:

- Creating and selecting a program
- Variable browsing
- Spreadsheet use

The supplied tutorial programs declare and initialize static and dynamic variables. Some of the variables are scalars, some are arrays, and some are records and structures.

The tutorial files are in the **`/usr/lib/NightProbe/Tutorial`** directory. Source listings of these files are in:

- "C Sample - c_sample.c" on page C-11
- "Ada Sample - ada_sample.a" on page C-9.

Creating and Selecting a Program

1. The source code for the sample program used in this tutorial, as well as the compiled and linked binary, can be found in the **`/usr/lib/NightProbe/Tutorial`** directory and are included at the end of this chapter for reference. The sample program contains C and Ada code (the latter compiled with the Concurrent MAXAda compiler).

2. Either copy the appropriate binary file for your system from /usr/lib/NightProbe/Tutorial, or copy the source files and compile the program:

   e.g.

   ```
   /usr/bin/uncompress -c \
         /usr/lib/NightProbe/Tutorial/ada_sample.linux.Z > ada_sample
   chmod 777 ada_sample
   ```

   - or -

   ```
   cp /usr/lib/NightProbe/Tutorial/ada_sample.a .
   cp /usr/lib/NightProbe/Tutorial/c_sample.c .

   cc -g -c c_sample.c
   PATH=$PATH:/usr/ada/bin
   a.mkenv -g
   a.intro ada_sample.a
   a.partition -create active ada_sample
   a.build ada_sample
   ```

3. Invoke NightProbe with the following command:

   ```
   /usr/bin/nprobe &
   ```

   NightProbe displays the Main window.

4. Invoke the sample program with the following command:

   ```
   ./ada_sample
   ```

5. In the Main window, right-click the Resources icon in the Session Overview Area and select the Add Program... menu option.

   NightProbe displays the Program window.

6. In the Program window, press the Select... button to the right of the PID text field and select the executing process `ada_sample` as shown in the list of processes associated with your user. Press OK.

7. In the Program window, press the Add button. You will see your program in the Resources list in the Main window.

Variable Browsing

The following sections provide an example of the use of the Item Browser window. For more information about the Item Browser window, see Chapter 11, "Item Browser Window".

Right-click the Probe Items icon in the Main window and select the Add Item from Program... menu option.

The Item Browser window appears and contains a single root item, `ada_sample`, representing our program resource. Expand that root item by clicking the control box to the left of the icon. Four new items will appear: Functions, Globals, Files, and Packages.

Adding a global C variable

- Expand the Globals list by clicking the control box to the left of the Globals icon.
- Select the `c_global_int` variable by clicking on its icon once
- Press the Add button
- The item now appears in the Probe Items list in the Main window and its color has turned to orange in the Item Browser to indicate it has been added.
- Collapse the Globals list by clicking on the control box to the left of the Globals icon

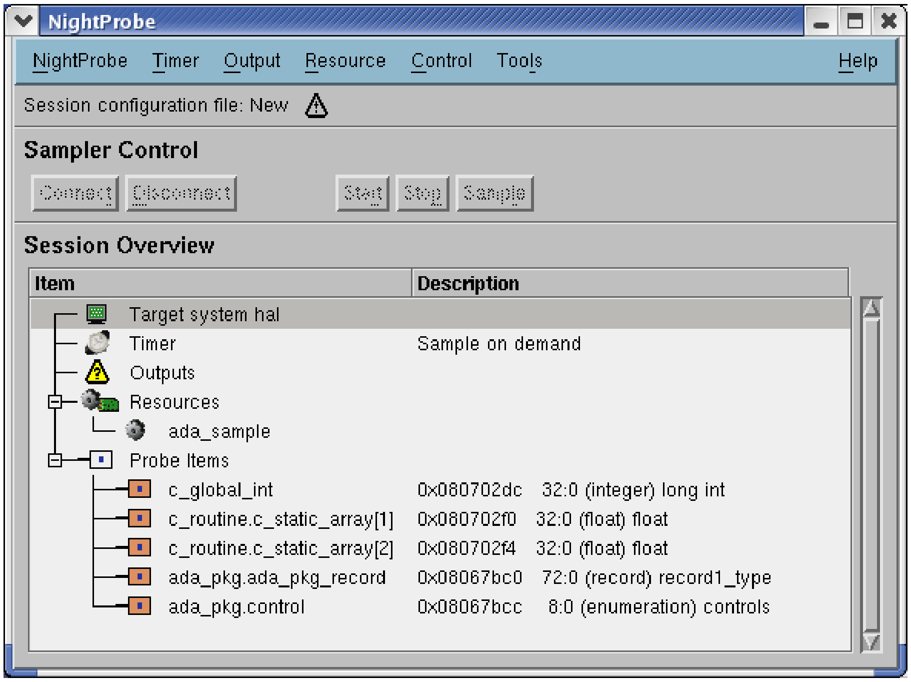
Adding a static C variable declared inside a function

- Expand the Functions list by clicking the control box to the left of the Functions icon.
- Expand the list of variables inside c_routine by clicking the control box to the left of the c_routine icon
- Expand the c_static_array component list by clicking the control box to the left of the c_static_array icon
- Select two components of the array by clicking on one, then clicking on the next one in the list while holding down the Shift key
- Press the Add button
- The two components now appears in the Probe Items list in the Main window and their color has turned to orange in the Item Browser to indicate they have been added.
- Click on the c_static_array icon once
- Now press the Left Arrow key on your keyboard until the Functions list is selected (if no action occurs, make sure that the c_static_array icon is selected and that your NumLock key is not on)
- Press the Spacebar to collapse the Functions list

Adding Ada variables declared in packages

- Now that you're an expert at navigating, browse the list of Packages until the variables in the package ada_pkg are visible
- Select the ada_pkg_record icon in the ada_pkg package
- Press the Add button to add the entire record to the list of Probe Items
- Add the control variable in the ada_pkg package as well
- Press the Done button to exit the dialog

At this point, the Main window should look like this:



Session configuration file: New

Sampler Control

Session Overview

Item	Description
Target system hal	
Timer	Sample on demand
Outputs	
Resources	
ada_sample	
Probe Items	
c_global_int	0x080702dc 32:0 (integer) long int
c_routine.c_static_array[1]	0x080702f0 32:0 (float) float
c_routine.c_static_array[2]	0x080702f4 32:0 (float) float
ada_pkg.ada_pkg_record	0x08067bc0 72:0 (record) record1_type
ada_pkg.control	0x08067bcc 8:0 (enumeration) controls

If the list of Probe Items differs from the figure, go back into the Item Browser and add the items shown above to the list.

NOTE:

The addresses to the right hand side of each item may differ from the figure above; that is expected and can be ignored.

Using the Spreadsheet

This section provides an example of the use of the Spreadsheet Viewer window. For more information about the Spreadsheet Viewer window, see "Spreadsheet Viewer" on page 14-1.

To launch the Spreadsheet Viewer, right-click the Outputs icon in the Session Overview area of the Main window and select the Spreadsheet Output menu option.

Quickly Adding Multiple Variables

1. In the Spreadsheet Viewer window, click on the uppermost left hand cell.

   The selected cell gets a black outline and an I-beam cursor.

2. From the Selected menu, select Place Variables.

   The Spreadsheet Variables window appears. All the variables from the Probe Items list in the Main window are shown in the list of variables in this window.

3. In the Spreadsheet Variables window, select all five entries by depressing and holding down the left mouse button on the top entry and dragging the cursor down through the last entry and then releasing the left mouse button. All five items should be highlighted.

4. Set the Cell Layout selection in the lower portion of the window to Vertical.

5. Set the Label Position selection to Left.

6. Set the Expansion selection to Both

7. Set the Direction selection to Vertical

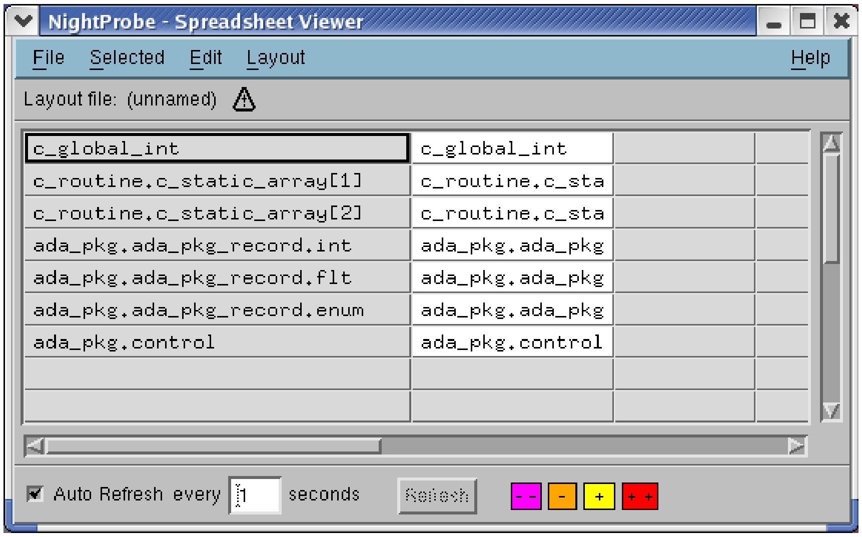
8. Press the OK button

   The spreadsheet cells starting with the uppermost left cell now describe the five variables you selected. The left hand column is a label field which includes the name of the variable. This field can be edited. The right hand column initially contains the same text, but will be replaced by the value of the associated variable when actual data sampling occurs.

   > Select the first cell column and use Column Width menu option from the Layout menu to widen the column to 30 characters

9. Select the second cell column and use Column Width menu option from the Layout menu to widen the column to 15 characters

At this point, the spreadsheet window should look like this:



Layout file: (unnamed) ⚠

c_global_int	c_global_int		
c_routine.c_static_array[1]	c_routine.c_sta		
c_routine.c_static_array[2]	c_routine.c_sta		
ada_pkg.ada_pkg_record.int	ada_pkg.ada_pkg		
ada_pkg.ada_pkg_record.flt	ada_pkg.ada_pkg		
ada_pkg.ada_pkg_record.enum	ada_pkg.ada_pkg		
ada_pkg.control	ada_pkg.control		

☑ Auto Refresh every 1 seconds

If your spreadsheet differs significantly from the above figure, select the cells with content, remove their content with the Cut menu option from the Edit menu and repeat the steps above.

Selecting a Timing Source

In the Main window, right-click the Timer icon in the Session Overview area and select the System Clock... menu option.

A System Clock Timing Source Configuration window will appear. Ensure the sampling rate is 1 second, and press the OK button.

Start Data Sampling

1. Connect to the target process by pressing the Connect button in the Sampler Control area of the Main window.

2. Begin sampling data by pressing the Start button in the Sampler Control area of the Main window.

   See the values of the variables displayed in the non-label cells of the Spreadsheet Viewer window. The `ada_sample` program changes the values of its variables once per second. Note that the frequency in which values are updated in the spreadsheet is actually unrelated to the frequency at which the sampling occurs; but by circumstance, they both currently happen to be 1 second. The bottom pane of the Spreadsheet Viewer window controls the frequency of spreadsheet refreshes. This is especially important in situations in which NightProbe is being used to record and log data to a file at high-frequency rates but at the same time is used interactively to peek and poke at variables.

Modifying the Value of Variables

Variables can be modified directly through the spreadsheet by typing in new values into their cells.

1. Modifying a variable

   a. Click the first data value cell associated with `c_global_int` in the 2nd column of the first row of the spreadsheet.

      The value moves to the left-hand side of the cell and stops updating but the remainder of the spreadsheet continues to be refreshed with data samples.

   b. Type in a new value for the cell by backspacing over the existing text and specifying 200. Press the Enter key.

      The value of `c_static_global` has been modified and is displayed. The program increments the value once per second.

2. Controlling execution of the program

   The main program uses the variable `ada_pkg.control` to control execution of the program in the following manner:

```
loop
   case ada_pkg.control is
   when halt   => exit program
   when run    => update variables
   when hold   => do nothing
   end case ;
   sleep 1 second
end loop ;
```

   a. Click on the cell showing the value of the control variable whose label is displayed in the cell to its left.

   b. Backspace over the existing value and type in "hold" (without the quotes) and press the Enter key

      This causes the program to skip updating variables -- notice how the values in the spreadsheet no longer change even though sampling is still active.

   c. Click on the same cell and change the value back to to "run" (without the quotes) which causes the program to resume updating variables.

   d. Click on the same cell and change the value to "halt" (without the quotes) which causes the program to exit.

NOTE

Look at the terminal screen where you invoked the `ada_sample` program; it should have exited.

Exit NightProbe by choosing the Exit menu option from the NightProbe menu in the Main window.

Ada Sample - ada_sample.a

```
package ada_pkg is
--
   type states is (none, init, freeze, start, stop, in_flight, approach, land);
   type controls is (halt, run, hold);

   type record1_type is
      record
         int   : integer := 0;
         flt   : float := 0.0;
         enum  : states := none;
      end record;

   ada_pkg_int    : integer := 0;
   ada_pkg_float  : float := 0.0;
   ada_pkg_enum   : states := none;
   ada_pkg_record : record1_type;

   control        : controls := halt;

   package nested is
      type array_type is array (1..4) of integer;
      type record2_type is
         record
            x : record1_type;
            y : array_type;
         end record;
      data : record2_type;
   end nested;
--
end ada_pkg;

with ada_pkg;

procedure ada_routine is
begin
--
   -- Increment variables in ada_pkg
   ada_pkg.ada_pkg_int := ada_pkg.ada_pkg_int + 1;
   ada_pkg.ada_pkg_float := ada_pkg.ada_pkg_float + 2.0;
   case ada_pkg.ada_pkg_enum is
   when ada_pkg.states'last =>
      ada_pkg.ada_pkg_enum := ada_pkg.states'first;
   when others =>
      ada_pkg.ada_pkg_enum := ada_pkg.states'succ(ada_pkg.ada_pkg_enum);
   end case;
   ada_pkg.ada_pkg_record.int := ada_pkg.ada_pkg_record.int + 1;
   ada_pkg.ada_pkg_record.flt := ada_pkg.ada_pkg_record.flt + 2.0;
   case ada_pkg.ada_pkg_enum is
   when ada_pkg.states'last =>
      ada_pkg.ada_pkg_record.enum := ada_pkg.states'first;
   when others =>
      ada_pkg.ada_pkg_record.enum :=
         ada_pkg.states'succ(ada_pkg.ada_pkg_record.enum);
   end case;

   -- Increment variables in nested_pkg in ada_pkg
   for i in ada_pkg.nested.data.y'range loop
      ada_pkg.nested.data.x.int := ada_pkg.nested.data.x.int + 1;
      ada_pkg.nested.data.y(i) :=
         ada_pkg.nested.data.y(i) + i;
   end loop;
```

```
--
end ada_routine;


with ada_routine;
with ada_pkg;

procedure ada_sample is
--
   procedure c_routine;
   pragma import (C, c_routine);
   pragma linker_options ("c_sample.o");
--
begin
--
   ada_pkg.control := ada_pkg.run;

   loop
      case ada_pkg.control is
      when ada_pkg.halt =>
         exit;
      when ada_pkg.run =>
         ada_routine;
         c_routine;
      when ada_pkg.hold =>
         null;
      end case;
      delay 1.0;
   end loop;
--
end ada_sample;
```

C Sample - c_sample.c

```
long            c_global_int = 1;

static long     c_static_int = 10;

void
c_routine(void)
{
   struct struct_type {
      int   int_component;
      float float_component;
   };

   static struct struct_type c_static_struct = {1, 2.0};
   static float              c_static_array[4] = {0.0,1.0,2.0,3.0};
   static int                c_static_func_int = 50;
   auto   int                c_stack_int = 0;
   auto   int                i;

   c_global_int ++;
   c_static_int += 2;
   c_static_func_int += 3;
   c_stack_int += 4;
   c_static_struct.int_component ++;
   c_static_struct.float_component += 1.1;

   for (i=0; i<4; i++) {
   if (c_static_array[i] > 10000.0) {
      c_static_array[i] /= 1.754;
   } else {
      c_static_array[i] *= 1.754;
   }
   }
}
```

Probing Devices Tutorial

This tutorial demonstrates NightProbe's ability to probe PCI devices.

This tutorial is only applicable to iHawk systems running RedHawk Linux.

However, other target resources, such as Shared Memory segments and Mapped Memory files may be probed on PowerMAX OS systems so reading the tutorial would be beneficial to PowerMAX OS users. For example, VME devices may be probed on PowerMAX OS system by using the Mapped Memory window and mapping /dev/mem with the appropriate physical address of the VME device.

We will probe the sync_clock timer on the Real-Time Clock and Interrupt Module (RCIM) which is present in all iHawk systems.

This tutorial requires that you run as the `root` user or that your user has the `CAP_SYS_RAW_IO` system capability as described in RedHawk User Capabilities in Appendix E.

Selecting the RCIM

We will use some type structures from a compiled program file to aid in viewing the RCIM device. Copy the rcim.c source file from /usr/lib/NightProbe/Tutorial/rcim.c and compile and link it in a working directory:

```
cp /usr/lib/NightProbe/Tutorial/rcim.c .

g++ -g -o rcim rcim.c
```

Invoke Nightprobe:

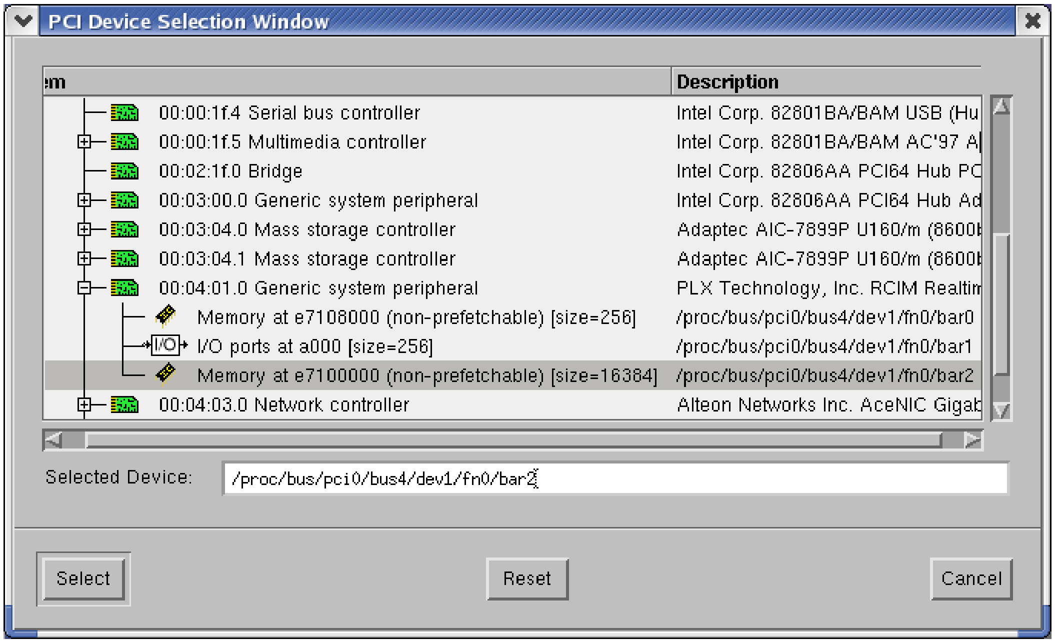
```
/usr/bin/nprobe &
```

1. Right-click the Resources icon and select the Add PCI Device... menu option in the Session Overview area of the Main window.
2. Press the Search... button on the dialog to launch the PCI Selection Dialog window
3. Scroll through the list of PCI devices until you see one labelled

   PLX Technology, Inc. RCIM Realtime Clock and Interrupt Module

4. Expand the list of regions for that device by clicking the control box to the left of the PCI card icon on that row.

The window contents should look similar to the following, with the actual set of PCI devices dependent on your specific system:



›m	Description
00:00:1f.4 Serial bus controller	Intel Corp. 82801BA/BAM USB (Hu
00:00:1f.5 Multimedia controller	Intel Corp. 82801BA/BAM AC'97 A
00:02:1f.0 Bridge	Intel Corp. 82806AA PCI64 Hub PC
00:03:00.0 Generic system peripheral	Intel Corp. 82806AA PCI64 Hub Ad
00:03:04.0 Mass storage controller	Adaptec AIC-7899P U160/m (8600
00:03:04.1 Mass storage controller	Adaptec AIC-7899P U160/m (8600
00:04:01.0 Generic system peripheral	PLX Technology, Inc. RCIM Realtim
Memory at e7108000 (non-prefetchable) [size=256]	/proc/bus/pci0/bus4/dev1/fn0/bar0
I/O ports at a000 [size=256]	/proc/bus/pci0/bus4/dev1/fn0/bar1
Memory at e7100000 (non-prefetchable) [size=16384]	/proc/bus/pci0/bus4/dev1/fn0/bar2
00:04:03.0 Network controller	Alteon Networks Inc. AceNIC Gigab

Selected Device: /proc/bus/pci0/bus4/dev1/fn0/bar2

5. Select the third region for the RCIM Device, the Memory region listed after the I/O ports by clicking it once

6. Press the Select button. The PCI Device Selection window will exit and the Vendor ID, Device ID, Region Index fields will be populated.

7. Change the Offset text field in the lower portion of the PCI Device window to the value 0x1000 (4096 decimal).

8. In the Symbol File text field of the PCI Device window, type in the name of the rcim program we built in the steps above

   ./rcim

9. Change the Resource Tag text field to `rcim` for clarity.

10. **<u>STOP.</u>. Make sure that the Offset field in step 7 is set to 0x1000. Proceed.**

11. Press the Add button. The selected PCI device memory region will now appear in the Resources list in the Session Overview area of the Main window.

Creating a View into the Device

In order to view locations within the PCI device, we need to create artificial variables to which we assign offsets and types. These variables are not allocated by NightProbe in the PCI device, they merely represent a view into a memory location that already exists in the device.

Right-click the PCI card icon in the Resources list in the Session Overview area of the Main window and select the New Item for PCI Device... menu option.

The Item Definition window appears. In the list area, you should see three root items representing Functions, Globals, and Files. The lists can be expanded so that you can browse for types to apply to the artificial variable you are creating within this dialog.

NOTE:

> If the only root item in the list is Globals, you forgot to specify a Symbol File value in the PCI Device window. Exit this dialog, right-click on the PCI card icon in the Main window, choose Properties..., and type in **`./rcim`** in the Symbol File text field and then return to this dialog.

- Expand the Files list by clicking the control box to the left of the Files icon.
- Expand the list of types inside "rcim.c" by clicking the control box to the left of the "rcim.c" icon
- Select the struct rcim_timer type by clicking on its icon once; do not expand the list of components.
- Change the Item Tag text field to the value sync_counter for clarity in subsequent displays
- Press the Add button

The artificial variable `sync_counter` has been added to the Probe Items list in the Session Overview area of the Main window.

Selecting the List Window

For brevity, we will select the simplest output method, the List Viewer window.

Right-click the Outputs icon in the Session Overview area of the Main window and select the List Window menu option.

A List Viewer window will appear and its textual contents will be empty.

Probing the RCIM

Press the Connect button in the Sampler Control area of the Main window.

NOTE:

If NightProbe pops up a diagnostic window that says you do not have sufficient permission to mmap the memory region, you need to either run as the `root` user or have your system administrator give you the `CAP_SYS_RAW_IO` capability as described in RedHawk User Capabilities in Appendix E. Save the current NightProbe session by selecting Save Session from the NightProbe menu in the Main window and then exit NightProbe. You can return to this spot in the tutorial as the `root` user or after you have been granted the required capability. Invoke **nprobe** with an argument which is the name of the session file you just saved.

At this time we have connected to the target system and mapped the memory region of the RCIM into the sampler process's address space. No sampling or probing has yet occurred.

For PCI device probing, it is usually best to use On Demand sampling since many devices may be sensitive to reads.

In the List Viewer window, a header has appeared:

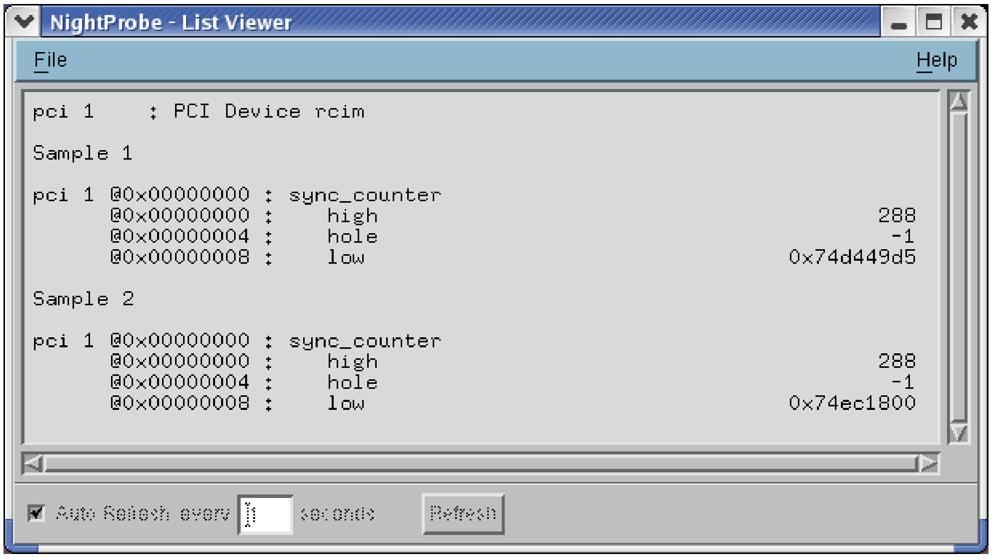
```
pci 1   : PCI Device rcim
```

Press the Sample button in the Sampler Control area of the Main window.

Press the Sample button again.

The List Viewer window will automatically display the contents of the artificial variable we created. The RCIM sync_clock timer has two portions, a high-order 32 value and a low-order 32 bit value separated by a 32 bit "hole".

The low-order word ticks once every 400 nanoseconds. The high-order word ticks once the low order word reaches 2**32 ticks.



```
pci 1     : PCI Device rcim

Sample 1

pci 1 @0x00000000 : sync_counter
      @0x00000000 :    high                              288
      @0x00000004 :    hole                               -1
      @0x00000008 :    low                        0x74d449d5

Sample 2

pci 1 @0x00000000 : sync_counter
      @0x00000000 :    high                              288
      @0x00000004 :    hole                               -1
      @0x00000008 :    low                        0x74ec1800
```

Exit NightProbe using the Exit menu option from the NightProbe menu of the Main window

C Source -- rcim.a

```
struct rcim_counter {
     int high;
     int hole;
     unsigned low;
};
```

struct rcim_counter counter;

main(){}

Conclusion

This concludes both tutorials. We hope that we have given you a sufficient overview so that you can get started using NightProbe. Reference the *NightProbe User's Guide* or use the context-sensitive help for the product if you have any questions.

D
GUI Customization

X Window System Resources

The graphical user interface (GUI) for NightProbe is based on OSF/Motif, and it runs in the environment of the X Window System. All X applications may be customized using X resources. Resources specify application attributes such as fonts, colors, screen layout, button and label names, mnemonics, accelerators, and text messages.

NightProbe provides default values for its X resources. Each user may override any X resources with personal preferences, or a site may provide for different defaults. These new settings can appear in the following places:

- In your **`.Xdefaults`** file
- On the **`nprobe`** invocation line (See "Command-Line Options" on page D-2.)
- In a resource file that the **`xrdb(1)`** X resource database manager reads

If you specify the same X resource on the **`nprobe`** invocation line and in your **`.Xdefaults`** file, the setting on the invocation line overrides the one in the file.

This appendix contains information that you need if you want to customize the graphical user interface.

NightProbe's behavior may be modified by specifying resources. Resources can be specified in many ways. One way to specify resources is to copy the default resource file to your home directory and change your version of NightProbe's resource file. That is the method used in this appendix. For more information on setting X11 client resources, refer to the *X Window System User's Guide*, to the *OSF/Motif Style Guide*, or to the **`X(1)`** and **`xrdb(1)`** man pages.

The following files in the **`/usr/lib/X11/app-defaults`** directory contain NightProbe's default resources:

`Nprobe`	Application default resources file

You can look in this file for examples of ways to customize NightProbe's appearance and behavior. To see all the NightProbe resources, use the **`editres(1)`** tool.

Command-Line Options

NightProbe has its own set of command-line options. (See "Invoking NightProbe" on page 2-1.) When you invoke NightProbe, you may also specify any standard X tool kit command-line option. Such options include **-bg** *color* to set the color for the window background; **-fg** *color* to set the color to use for text or graphics; and **-xrm** *resourcestring* to set selected resources. For example:

```
nprobe -xrm "*drecWindow*geometry:-0+0" &
```

would put the Main window in the upper right corner. For a complete list of these options, refer to the **X(1)** man page.

Application Resources

In addition to the standard resources associated with an X11 or Motif program, NightProbe defines special *application resources* you can use to customize NightProbe's appearance and behavior. These resources affect the entire NightProbe graphical user interface; they are "global" to the application.

There are two categories of application resources used by NightProbe. One set of application resources applies to all products that are part of the NightStar tool set. In addition to these, NightProbe has its own application resources.

NightStar Resources

NightProbe is part of the NightStar tool set. To provide a consistent appearance among these tools and to provide an easy way for you to change the default appearance, special application resources exist that define fonts and colors. They allow you to change one resource (instead of many) to affect the font or color for a set of window components that have similar characteristics. These resources are applied only to certain window components; many of NightProbe's window components are unaffected by the NightStar resources.

For example, some textual display areas show only program output and some areas accept input only from you. Different colors are used for these areas to distinguish them. If you want to change the color for input fields, for example, you need to change only one resource in NightProbe's resource file. See "Color Resources" on page D-4. The next time you run NightProbe, the color of all the input fields has the new setting.

Changing the inputBackground line in your **Nprobe** file to:

```
*inputBackground:         Yellow
```

causes the background color for all input areas to be yellow.

NightProbe Resources

NightProbe resources are not shared by other NightStar tools. A list of NightProbe resources follows.

`lowCautionColor` Control the color of the Low Caution field of the Spreadsheet Viewer window. The default value is orange. (See “Control Area” on page 14-10.)

`lowDangerColor` Control the color of the Low Danger field of the Spreadsheet Viewer window. The default value is magenta. (See “Control Area” on page 14-10.)

`highCautionColor` Control the color of the High Caution field of the Spreadsheet Viewer window. The default value is yellow. (See “Control Area” on page 14-10.)

`highDangerColor` Control the color of the High Danger field of the Spreadsheet Viewer window. The default value is red. (See “Control Area” on page 14-10.)

`text.maxArrayExpansion` Control the maximum number of array components which can are shown when expanding an array variable or type in the Item Browser and Item Definition windows. The default value is 1000 components. (See “Interactive Variable Browser” on page 11-2.)

Font Resources

Your X terminal vendor supplies you with vendor-specific directories and files that pertain to fonts. The programs **`xlsfonts(1)`** and **`xfontsel(1)`** can be used to help you find font names. NightProbe’s font resources are in the file **`/usr/lib/X11/app-defaults/Nprobe`**. This section describes the special font resources available for NightStar tools in general and NightProbe in particular.

NightStar tools use proportional-width fonts except in areas that depend on text alignment; in these instances a fixed-width font is important for readability. If you decide to change fonts, make sure that you choose another fixed-width font for the font resources that have `fixed` in their names.

NightStar font resources include:

`smallFontList` Used for areas that require a smaller font. NightProbe does not currently use this font.

`infoFontList` Used for areas that display informational messages, warnings, errors. NightProbe uses this font for text fields.

`fixedFontList` Used for areas that depend on text alignment. NightProbe uses this font for lists and the viewing area of the Spreadsheet Viewer window.

`smallFixedFontList` Used for areas that depend on text alignment but require a smaller font. NightProbe does not currently use this font.

NightProbe uses a *default font* for most of the textual display in the windows. This proportional-width font is specified as the value of the standard Motif `fontList` resource. This font is used by window components that do not have a font specified for them. For example, changing the `fontList` line in your **`Nprobe`** file to:

```
*fontList:          9x15
```

would cause NightProbe to use the `9x15` font for the default font of most textual displays.

Color Resources

This section describes the special color resources available for NightStar tools in general and NightProbe in particular.

NightStar tools use the same color scheme to indicate that they are part of the same tool set and to provide cues about the usage of different areas in the windows. Each NightStar tool uses a unique color for its menu bars.

The following NightStar color application resources are defined:

`outputBackground` `outputForeground`	Used for the background and foreground colors in output-only areas.
`inputBackground` `inputForeground`	Used for the background and foreground colors in areas that accept user input.
`distinctBackground` `distinctForeground`	Used for the background and foreground in colors areas that require user input.

NightProbe uses a *default color* for most of the window areas. This color is specified as the value of the standard X11 `background` resource. This color is used by window components that do not have a color specified for them.

E
Target System Requirements

This chapter provides an overview of the user and system configuration requirements that need to be taken into account prior to running NightProbe on a target system.

RedHawk User Capabilities

If you wish to probe PCI devices or other target resources to which you do not have appropriate file access (e.g. **/dev/mem** or a shared memory segment or process owned by a different user), you must have the CAP_SYS_RAWIO capability.

Have your system administrator register you as a NightProbe user according to the following instructions:

1. Add the following line to the ROLES section of the **/etc/security/capability.conf** file if the role is not already present:

   ```
   role probeuser cap_sys_rawio
   ```

2. Add the folowing line to the bottom of the **/etc/security/capability.conf** file:

   ```
   user username probeuser
   ```

   where *username* is your login name.

See **pam_capability(8)** for more information.

PowerMAX OS System Configuration

The default kernel configuration shipped with the system should be sufficient for NightProbe use. However, the system administrator should be aware of the following kernel options.

Table C-1 describes the kernel options that NightProbe requires.

Table C-1. Required Kernel Options

Kernel Option	Description
`fbs`	Frequency-based scheduler. Required only if you use the frequency-based scheduler in conjunction with NightProbe.
`ipc`	Inter-process communications, including shared memory. For more information about shared memory, see "Interprocess Communication" in the *PowerMAX OS Programming Guide*.
`procfs`	**`/proc`** file system.

See the *NightSim User's Guide* for additional information about establishing the environment for NightSim, an optional real-time tool that provides access to the frequency-based scheduler and performance monitor.

NightProbe can produce information that can be analyzed by the NightTrace performance analysis tool. See the *NightTrace Manual* for information about establishing the environment for NightTrace.

Index

Spine for 1/2” Binder

Product Name: 0.5" from top of spine, Helvetica, 36 pt, Bold

Volume Number (if any): Helvetica, 24 pt, Bold

Volume Name (if any): Helvetica, 18 pt, Bold

Manual Title(s): Helvetica, 10 pt, Bold, centered vertically within space above bar, double space between each title

Bar: 1" x 1/8" beginning 1/4" in from either side

Part Number: Helvetica, 6 pt, centered, 1/8" up

NightProbe

User’s
Guide

0890465

Copy the contents of this chapter into the Table of Contents.

Illustrations

Do not include this document in the final book.

Copy the contents of this chapter into the Table of Contents.

Do not include this document in the final book.

Copy the contents of this chapter into the Table of Contents.

Tables

Do not include this document in the final book.

Copy the contents of this chapter into the Table of Contents.

Do not include this document in the final book.